MCAD – ECAD INTEGRATION

CONSTRAINT MODELING AND PROPAGATION

A Thesis
Presented to
The Academic Faculty

By

Kenway Chen

In Partial Fulfillment
Of the Requirements for the Degree of
Master of Science in Mechanical Engineering

Georgia Institute of Technology
December 2008

MCAD – ECAD INTEGRATION

CONSTRAINT MODELING AND PROPAGATION

Approved by:

Dr. Dirk Schaefer, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Jitesh Panchal
School of Mechanical and
Materials Engineering
Washington State University

Dr. David Rosen
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Chris Paredis
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Douglas Yoder
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: October 8, 2008

ACKNOWLEDGEMENTS

My mother told me that I was named Kenway because people with this name are excellent in analyzing, understanding, and learning. They are usually quiet and introspective and they will see the larger picture when presented with issues. They tend to be philosophers, scholars, and teachers. My parents picked the right name for me because it certainly describes my characteristics and personality. And to become a scholar and a researcher has always been my career goal.

I must first acknowledge my mother because she has played the most important role in my life. Since the passing of my father twenty years ago, my mother Alice has endured all sorts of hardship as a single-mother to raise me and guide me to a successful path. During the several years of staying in New York City, where I received both my junior high and high school education, my mother would ignore the extreme fatigue from long hours of work to ensure that I am healthy and on-track with my schoolwork, night after night. My acceptance to my dream school, UC Berkeley, gave my mother extreme pride but also further hardship. Despite having the tuition covered by fellowships, scholarships, and loans, the expense for living in the Bay Area for four years nonetheless exhausted much of my mother's financial resources. As a Graduate Research Assistant here at Georgia Tech I can support my living with my monthly stipend, however, I can yet to support my mother. Soon, this will all change. Once I start my position as a mechanical engineer at the National Institute of Standards and Technology, I should have the financial capability to allow my mother to live the worry-free living that she long deserved.

I must also acknowledge all the people that guided me in my graduate research. First I must thank my advisor Dr. Dirk Schaefer for all his words of encouragement, advices on time management, and of course, his valuable insights and guidance on my research work. I also thank Dr. Jitesh Panchal for spending considerable amount of time providing valuable research insights and strategies, guidance in writing conference papers and book chapter, and reading through my works to provide me valuable feedback. I must also thank my thesis committee, Dr. David Rosen, Dr. Chris Paredis, and Dr. Douglas Yoder, for their interest, intellectual contribution, and insightful feedback.

I must acknowledge my academic family in the Systems Realization Laboratory for the motivation in my graduate research and the enjoyment in my social life. I especially want to thank my colleagues at the Savannah campus. First of all, I have to thank my former roommate and colleague Nathan Young. He has been a true friend and provided me with significant amount of guidance and motivation. I also have to thank Jonathan Bankston for all the stimulating discussion we had regarding our research, and also his assistance in software problems I encountered. Finally I thank members of SRL Savannah, Alex Ruderman, Chenjie Wang, Markus Rippel, and Timothy Dietz for source knowledge, motivations, and friendship.

Finally, I acknowledge the support of the George W. Woodruff School of Mechanical Engineering as well as that of EPLAN Software and Services LLC for providing me with the ECAD/ECAE system EPLAN Electric P8.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Mechatronic systems encompass a wide range of disciplines, including mechanical and electrical engineering, and hence the development process for mechatronic system is collaborative in nature. Currently the collaborative development of mechatronic systems is inefficient and error-prone because contemporary design environments do not allow sufficient information flow of design and manufacturing data across different engineering domains. Mechatronic systems need to be designed in an integrated fashion allowing designers from multiple engineering domains to receive updates regarding design modifications throughout the design process. One approach to facilitate integrated design of mechatronic systems is to integrate mechanical with electrical engineering CAD systems.

Currently there exist numerous techniques that were developed to support various levels of integration between CAD/CAE systems. Standardized data exchange formats, e.g., STEP and IGES, support information exchange between various different CAD and PDM systems. Multi-Representation Architecture (Peak et al.) supports the integration of geometric information in CAD tools with analysis information in CAE tools. Other integration techniques include the Core Product Model (developed at NIST), Active Semantic Networks (Roller et al.), Constraint Linking Bridge (Kleiner et al.), and others. All these techniques have their areas of focus as well as research gaps that need to be covered. One area that needs research attention is the information exchange between mechanical and electrical domains, which is the focus of this thesis.

In this thesis, the information exchange between mechanical and electrical domains is explored from two perspectives: conceptual design perspective, in which constraint relationship between attributes of mechanical and electrical components is identified and classified based on the physical forms, functions, and behavior of the mechatronic system; system realization perspective, in which the identified constraints are modeled for propagation between MCAD and ECAD systems. SysML is used to model the constraints between mechanical and electrical components. By means of an illustrative example (a robot arm), the constraint modeling and propagation developed in my thesis are demonstrated and implemented utilizing a MCAD system (SolidWorks) and an ECAD system (EPLAN Electric P8).

CHAPTER 1
MOTIVATION FOR MCAD – ECAD INTEGRATION AND AN APPROACH TO INTEGRATED DESIGN OF MECHATRONIC SYSTEMS

Traditional mechanical systems were built using mechanical components only. In the steam engine designed by James Watt, the engine converts the steam energy into kinetic energy through the rotational motion of a shaft using a reciprocating mechanism. The speed of steam flow was controlled by a fly-ball governor. This is an example of a mechanical feedback system. In the twentieth century, electrical energy and electric signals were made available for industrial applications. These applications are more efficient using conversion of electrical energy into mechanical energy because electrical energy is easier to process as signals for measurement and control.

In the last few decades, digital computers have been included in most analog devices. Faster and more accurate results are obtained using a digital computer. As a result, most of today's products are a mix of mechanical, electrical, and digital components. Mechatronics involve the design of systems that incorporate a range of engineering disciplines.

Cross-disciplinary integration of mechanical, electrical, and electronic engineering as well as recent advances in information engineering are becoming more and more crucial for future collaborative design, manufacture, and maintenance of a wide range of engineering products and processes. In order to allow for additional synergy effects in collaborative product creation, designers from all disciplines involved need to adopt new approaches to design, which facilitate concurrent cross-disciplinary

collaboration in an integrated fashion. This, in particular, holds true for the concurrent design of mechatronic systems.

1.1 Mechatronic Systems: Context and Scope

Mechatronic systems usually encompass mechanical, electrical, electronic, and software components (see Figure 1.1). The design of mechanical components requires a sound understanding of core mechanical engineering subjects, including mechanical devices and engineering mechanics (Appuu Kuttan 2007). For example, expertise regarding lubricants, heat transfer, vibrations, and fluid mechanics are only a few aspects to be considered for the design of most mechatronic systems. Mechanical devices include simple latches, locks, ratchets, gear drives, and wedge devices as well as complex devices such as harmonic drives and crank mechanisms. Engineering mechanics is concerned with the kinematics and dynamics of machine elements. Kinematics determines the position, velocity, and acceleration of machine links. Kinematic analysis is used to determine the impact and jerk on a machine element. Dynamic analysis is used to determine torque and force required for the motion of link in a mechanism. In dynamic analysis, friction and inertia play an important role.

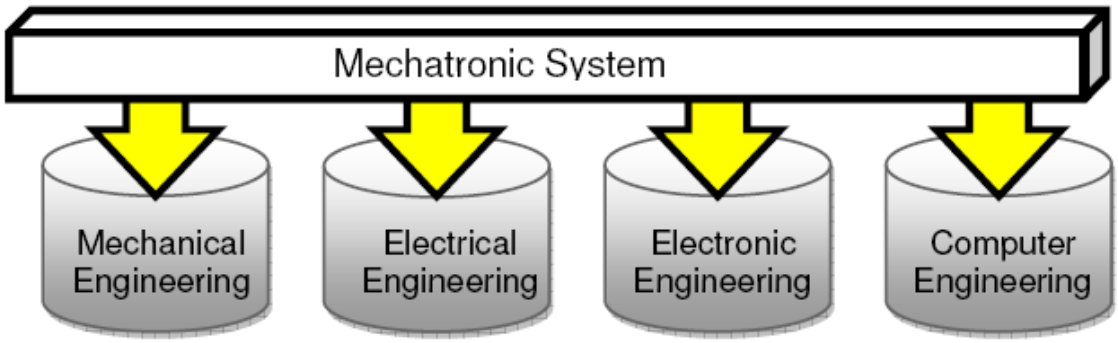


Figure 1.1: The scope of mechatronic system

Electronics involve measurement systems, actuators, and power control. Measurement systems in general comprise of three elements: sensors, signal conditioner, and display units. A sensor responds to the quantity being measured from the given electrical signal, a signal conditioner takes the signal from the sensor and manipulates it into conditions suitable for display, and in the display unit the output from the signal conditioner is displayed. Actuation systems comprise the elements which are responsible for transforming the output from the control system into the controlling action of a machine or a device. And finally power electronic devices are important in the control of power-operated devices. The silicon controlled rectifier is an example of a power electronic device which is used to control DC motor drives.

The electrical aspect of mechatronic systems involves the functional design of electrical plants and control units. This is done through the generation of several types of schematics such as wiring diagrams and ladder diagrams. In addition, programmable logic controllers (PLCs) are widely used as control units for mechatronic systems (see Figure 1.2). PLCs are well-adapted to a range of automation tasks. These typically are industrial processes in manufacturing where the cost of developing and maintaining an automation system is relatively high compared to the total cost of the automation, and where changes to the automation system are expected during its operation life.

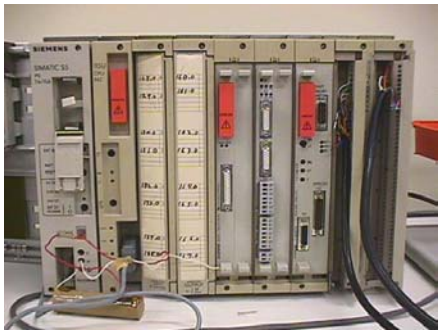


Figure 1.2: An example of programmable logic controllers

1.2 Motivation for MCAD – ECAD Integration

In designing a mechatronic system, there are many situations that require the exchange of information between CAD models of different domains, such as Mechanical Computer-Aided Design (MCAD) and Electrical Computer-Aided Design (ECAD) systems, or between CAD systems and other Computer-Aided Engineering applications (Chen and Schaefer 2007). For example, a company may use different CAD systems for different purposes along the product design process (see Figure 1.3) or companies and their suppliers may use different CAD systems to collaborate.

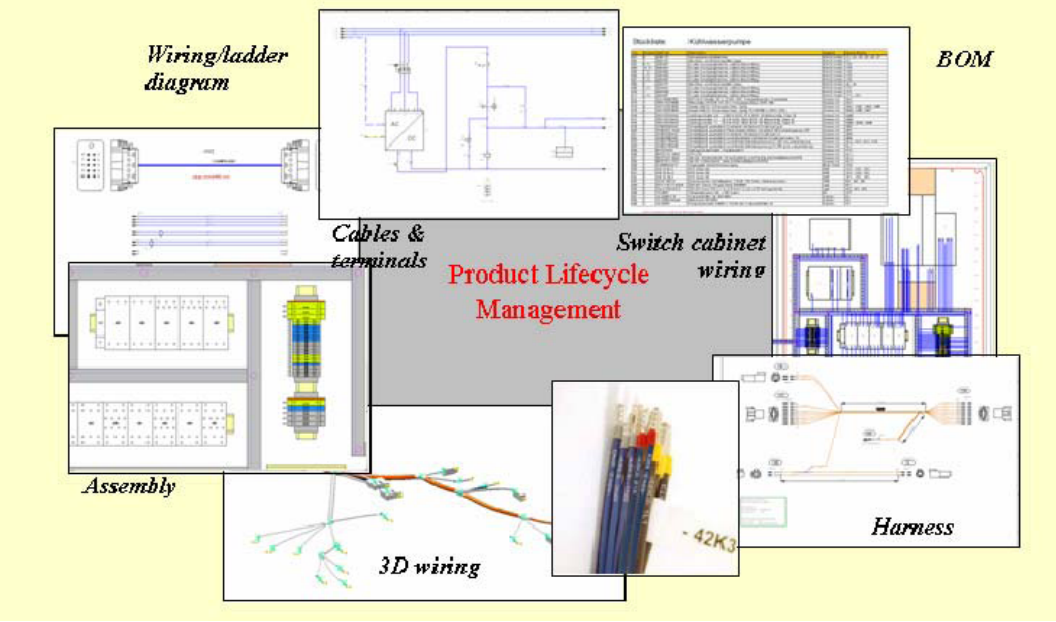


Figure 1.3: Various ECAD-specific application domains (Schaefer 2006)

During the design process of mechatronic system, modifications made on a MCAD model may lead to significant design modifications to be made on the corresponding ECAD model and vice versa. Obviously, there exist a huge number of constraints between a mechanical part of a mechatronic product and its electrical counterpart that have to be fulfilled to have a valid design configuration. As of yet, such

interdisciplinary constraints between models of different engineering design domains cannot be handled in a multidisciplinary computer-aided engineering environment due to the lack of appropriate multidisciplinary data models (Roller et al. 2002).

For companies that design mechatronic systems, staying competitive in today's market means using design systems that unify the design process and allow the a smooth flow of design data across the disciplinary divide, such as the electro-mechanical divide. As the mechatronic systems and the processes of creating them evolve, the "*fundamentally dissimilar worlds of electronic [and electrical] and mechanical design need to work in harmony*." (Evans 2007).

Today, due to a lack of integration between MCAD and ECAD systems and the fact that design engineers of either domain (electrical and mechanical) usually are not familiar with design consequences that their decisions may induce at the other domain or counterpart systems, design consequences are often noticed at later stages of the product design process. In other words, designers and engineers do not know the effect of a particular modification on a design model until that modification has been implemented. If the corresponding modification on the counterpart model cannot be implemented, designers must track back to the modification and reconsider alternate options, and such process leads to additional revision cycles, time penalties, and associated cost.

In order to improve the overall design process, these design consequences have to be identified and considered at an earlier time. This, however, would require the electrical and mechanical CAD systems to bi-directional exchange data to allow designers in both domains to cooperate more efficiently. Benefits of the integration of electrical and mechanical CAD systems include:

- Reduced product time-to-market.
- Designers' awareness of the overall design process.
- Improved quality of products.
- Increased company productivity.

The product time-to-market is reduced because the designers are aware of the design consequences in the early stages of the design process and therefore avoid implementing undesired modifications. The designers are aware of the overall design process because the direct communication between MCAD and ECAD systems allows them to posses the knowledge of design consequences from both the electrical and mechanical side. The quality of the product is improved because the designers no longer have to spend time considering unnecessary modifications and re-modifications. This means that more time will be dedicated to maintaining quality and satisfying customer specifications. With all the above advantages combined, achieving MCAD – ECAD integration will increase a company's productivity.

1.3 Research Vision

As alluded to in the previous section, achieving MCAD – ECAD integration will lead to benefits such as reducing product time-to-market and increasing company productivity. Furthermore, one application as a result of such cross-disciplinary integration will be a fully automated or in the early stage, a partially automated creation of complete technical product documentations. Such product documentations include CAD drawings, assembly plans, sub-assembly configurations, bill of materials, cable lists, terminal lists, wiring diagrams, etc. Automated creation of product documentation means

that documents such as the ones described above are automatically derived from the MCAD and ECAD models of a mechatronic system.

Today, such an automated creation of product documentations is possible on both MCAD and ECAD levels. In other words, many current MCAD and ECAD software packages have the capability to automatically generate assembly configuration, bill of materials, and all other diagrams based on their underlying data models. However, this is only at the level of individual domains, not in an integrated fashion. In other words, not only the assembly configuration, bill of materials, and all other diagrams of mechanical domains need to be updated, but also all diagrams in the electrical domains. With the integration of MCAD and ECAD software, the software would have the capability to automatically update all relevant parts of the product documentations in both domains whenever a design modification occurs (or whenever an updated version of the product documentation is needed).

Mechatronic systems encompass not only electrical and mechanical engineering, but also other domains including electronic and software engineering. Hence the ultimate dream is to achieve integration of all domains relevant to the design of mechatronic systems. For example, a mechatronic design team may be designing an elevator. They receive a customer request to increase the elevator capacity. By specifying the new elevator capacity, all relevant CAD models would be updated with necessary changes such as increase the size of the elevator and select a more powerful motor that supports the new elevator load, sub-assembly and assembly configuration is redrawn ensuring that the new motor fits into the cabinet box, bill of materials is updated, and cable list drawings and interconnection diagrams are revised to match the specification of the new

motor, and so on. In my dream I envisage a fully automated creation of product documentations covering all domains of the mechatronic systems.

This dream of a fully integrated and automated approach that includes all domains involved in the design of mechatronic systems cannot be achieved in one single step. I believe that it has to be done step-by-step through a horizontal ladder and a vertical ladder (see Figure 1.4). The horizontal ladder is the number of domains involved. It is difficult to integrate multiple domains all at once because with each additional domain, the amount and the complexity of information flow increase as the integrated system tries to incorporate the knowledge from that added domain. Hence the first step is to start with the integration of two domains. For example, the integration of mechanical and electrical domains and integration of mechanical and electronic domains should be achieved before attempting to achieve integration of mechanical, electrical, and electronic domains.

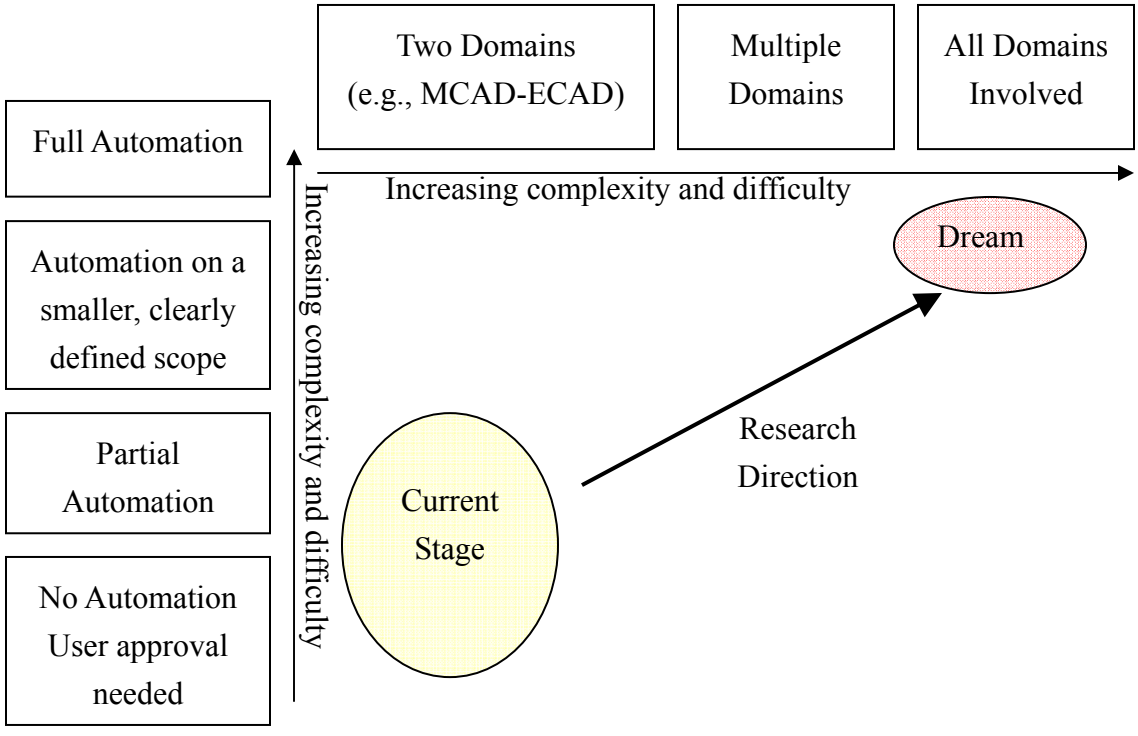


Figure 1.4: The Big Picture

The vertical ladder is the level of automation. The top level is the dream of fully automated creation of product documentation for all domains involved in the mechatronic design. At this level, the computer systems are required to possess the knowledge of experienced engineers to make appropriate decisions for carrying out full automation, which means complex and advanced expertise in computer science, expert systems, and artificial intelligence is required, therefore this level is difficult to achieve. Going one level lower, the fully automated creation of product documentation is carried out on a small, clearly defined context and scope. At this level, the designers and CAD software focus on component level of the mechatronic system rather than an entire mechatronic system.

For example, many mechatronic systems need to draw electrical power from a wall outlet, which means a power plug is needed. The power plug can vary from country to country as different countries use different settings for the power outlet. Automation at this level would mean users specify a country and CAD systems generate the correct CAD model of the power plug and the product documentation (bill of material, etc.) is automatically updated. It is still difficult to achieve this level as CAD systems would still need to possess the knowledge needed to make appropriate decisions to achieve full automation. For instance, the power plug is suitable for use in this country, but is it appropriate for this particular mechatronic system? At the third level, the objective is to achieve partial automation in which CAD systems can make some changes to their underlying CAD models in a clearly defined context and update the product documentation. At the fourth and lowest level, CAD systems cannot make changes automatically and humans have to make approvals for changes to occur.

In this thesis I address the first level of horizontal ladder and third and fourth level of the vertical ladder (see Figure 1.5). I address the integration of mechanical and electrical engineering domains. And I address automation on the scale of propagation of changes made to CAD model with both partial automation and involvement of human approving the changes. When a design modification occurs, the user is notified about the change and is given recommended changes that need to be made to the CAD model (e.g., dimension of a mechanical component, electrical specification such as maximum torque of a motor). The user is given the option of accepting or rejecting the change; this is the part where human approves the changes. If the change is accepted, the change (whether it is a dimension change on the MCAD side or an electrical component replacement on the ECAD side) is carried out automatically, which is the partial automation part.

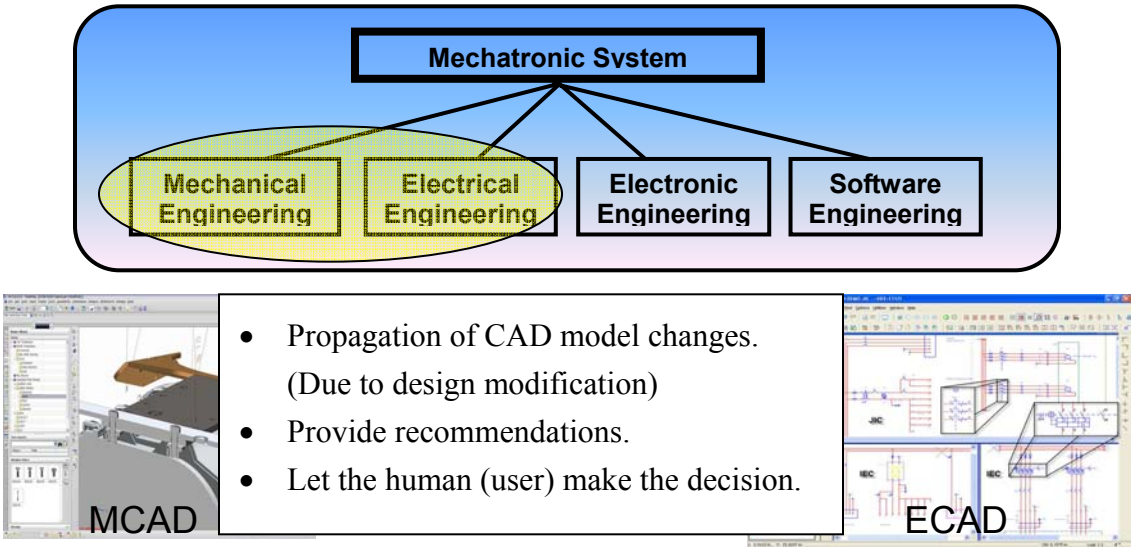


Figure 1.5: The Research Focus

1.4 Research Questions and Hypotheses

The research questions and hypotheses aim at achieving the stage described in the big picture: integration of MCAD and ECAD at partial automation level.

The scope of this work is to address the propagation of CAD model changes on both conceptual design level and CAD system implementation level, through the approach of constraint modeling and propagation. While the software program for achieving the propagation cannot be implemented until the CAD models of the mechatronic system have been created because information from the CAD models are needed (e.g., the name of the part), the model for driving this propagation can be created during the conceptual design stage because most of the design constraints would have been specified at that point.

Given this context, the research question and its hypothesis are as follows:

Research Question

How can mechatronic systems be designed in an integrated fashion in order for designers of both electrical and mechanical engineering domains to automatically receive feedback regarding design modifications made on either side throughout the design process?

Hypothesis

Integrated design of mechatronic systems can be realized through the integration of mechanical and electrical CAD systems. One approach to achieve this type of integration is through the propagation of constraints. Cross-disciplinary constraints between mechanical and electrical design domains can be classified, represented, modeled, and bi-directionally propagated in order to provide automatically feedback to designers of both engineering domains.

There are three phases toward achieving the answer to this Research Question. The first two phases address the identification of design constraints and creation of a constraint model, which is at conceptual design level. The third phase addresses propagation of CAD model changes, which is at CAD implementation level.

The first phase is identification of constraints followed by a classification of these constraints. This is important because before the discussion regarding constraint modeling and propagation can take place, one needs to know the exact meaning of a constraint, and examples of constraints in a mechatronic system. The first phase of this research is constraint identification and classification. This is the focus of the first Sub-Research Question.

Sub-Research Question 1

How can discipline-specific design constraints in both mechanical and electrical engineering domains be classified?

Sub-Hypothesis 1

Design constraints in both domains can be classified through analyzing mechatronic systems to identify and classify discipline-specific constraints based on associated function, physical form, and system behavior. Constraints based on functions involve the use of equations, constraints based on form involve observation of the physical appearance, and constraints based on behavior involve analyzing system response. These constraints allow direct mapping of design aspects between mechanical and electrical domains of the mechatronic system.

The purpose of classifying constraints based on functions, forms, and behavior is to achieve a direct mapping of design aspects between mechanical and electrical domains. Much research has been done on constraint modeling on the mechanical engineering side. However, the search for connection of mechanical constraints to the electrical side, and vice versa, is still an unrevealed research territory. Hence this Sub-Research Question 1 and Sub-Hypothesis is an attempt to establish fundamental theories in this research area.

The second phase is modeling of the constraints that are identified and classified in the first phase. In order to implement constraint propagation between electrical and mechanical domains, a model that is to be used as the foundation for the implementation is needed. In addition to serving as the foundation for the implementation, the constraint model provides a conceptual view of the constraint propagation. From the model one can identify the direct mapping between the characteristics or attribute values of the components in a mechatronic system. The second phase of this research is constraint modeling. This is the focus of the second Sub-Research Question.

Sub-Research Question 2

How can both discipline-specific and cross-disciplinary constraints be modeled?

Sub-Hypothesis 2

Both discipline-specific constraints and cross-disciplinary constraints can be modeled using SysML. SysML supports the specification and analysis aspects, the constraints in the mechatronic system are well represented using SysML's Parametric Diagram.

The purpose of a constraint model is for usage in the implementation to achieve communication between mechanical and electrical domain, in particular, between MCAD systems and ECAD systems. Two object-oriented modeling languages appear as candidates for creating such constraint model: UML and SysML. UML is suitable for modeling various aspects of a mechatronic system but lacks the capability to model engineering analysis. SysML, on the other hand, covers both aspects and hence is identified as the better approach of the two for creating a constraint model for mechatronic systems. The Sub-Research and Sub-Hypothesis aim at creating a model that addresses the various aspects of a mechatronic system (e.g., characteristics or state of a component) and parametrics for use in constraint propagation.

The third phase is the development of constraint propagation based on the model created in the second phase. The constraint model provides a means of expressing parameters and relationships between parameters in forms of equation. The implementation is capable of retrieving data from software systems, carry out the calculation, and notify the other side of the results. The implementation is also capable of sending a request, recording the response, and sending the response back to the sender. The third phase of this research is constraint propagation. This is the focus of the Development Question.

Development Question
How can constraint propagation be implemented in SolidWorks and EPLAN?

Development Hypothesis
With the macro capability of both CAD systems (MCAD and ECAD), component configuration can be defined with given attribute ranges. A constraint manager can be developed to take the new attribute value resulting from a design modification, calculate the new corresponding attribute value on the other domain based on the predefined constraint relationship in the constraint model, and update the designers.

The implementation satisfies the various scenarios that are possible during the constraint propagation process, such as new attribute values exceeding the limitation of the design, or the designers for whatever reason cannot accept such modification. Each scenario is specified in a diagram called "*constraint propagation flow diagram*". The implementation involves information flow between SolidWorks and EPLAN, the Application Programming Interface of the CAD software, and Excel. The implementation is designed to take into account all the scenarios specified by the constraint propagation flow diagram.

1.5 Organization of the Thesis

The organization of this thesis is as follows: in Chapter 2, an overview of current techniques aimed toward achieving integrated design of mechatronic systems as well as a research gap analysis is provided. There are several approaches to support the information exchange across different engineering domains. Product data management systems manage product information from design to manufacture to end-user support. Standard data exchange formats such as IGES and STEP are developed to achieving communication between various CAD, CAE, and PDM systems. NIST developed a product information modeling framework that includes the core product model (CPM) to expand the support of production information and information exchange to a full range. Peak et al. developed multi-representation architecture (MRA) to achieving information mapping between design models and analysis models in CAD and CAE systems. And there are constraint-based approaches such as Colibri that focuses on the parametric relations among the CAD/CAE models.

To address Sub-Research Question 1, Chapter 3 begins with an overview of mechanical components and electrical components that are commonly used in mechatronic system. Then a list of different mechanical and electrical constraints is provided. This is followed by a discussion of constraint classification based on function, form and behavior. Towards the end of the chapter an illustrative example of a robot arm for presenting various types of constraints is provided.

Chapter 4 discusses constraint modeling and addresses Sub-Research Question 2. At the beginning of the chapter two object-oriented modeling languages are introduced: UML and SysML. A comparison between the two modeling languages is made and

SysML is chosen as the better candidate for use in constraint modeling of mechatronic systems because of its capability to perform engineering analysis. Brief description of using blocks and parametrics in SysML is provided in this chapter. Toward the end of this chapter, a constraint model that is developed for modeling components and attributes in mechatronic systems is introduced. An illustrative example of modeling a robot arm using this particular constraint model is shown in the last part of this chapter.

Chapter 5 discusses constraint propagation and addresses the Development Question. At the start of the chapter, a constraint propagation flow diagram is introduced. This flow diagram outlines the possible scenarios for propagating a design modification. This is followed by a discussion of the implementation of my constraint propagation approach. In the remainder of the chapter, test scenarios based on the constraint propagation flow diagram are provided to validate the implementation.

Chapter 6 is the closing chapter. At first, the hypotheses stated at the beginning of this thesis are evaluated and validated. This is followed by a discussion of the research contributions made and a discussion of the limitations of this research work. Finally future work directions are given for further development of this research work.

CHAPTER 2
STATE-OF-THE-ART AND RESEARCH GAPS

Constraint modeling and propagation is one method to facilitate integration of MCAD and ECAD systems. There are numerous other research activities, though not particularly focused on the integration of MCAD and ECAD systems, but are aimed at a broader common goal of information management and systems integration. A literature review was conducted to evaluate various documented research activities and state-of-the-art technologies regarding information management and systems integration. In Table 2.1 is a list of the major approaches to information management and systems integration that were evaluated in this literature review. Table 2.1 shows the name of approach and some of the corresponding literature, a brief description of the approach, and short summary of research gaps that are identified.

The mindset for conducting this literature review was to obtain a better understanding of how other researchers have tried to approach related problems and where the current research gap are.

The next several sections of this chapter contain the discussion of the various information management and systems integration approaches, including all the ones listed in Table 2.1. Toward the end of this chapter is the evaluation of each approach and research gap analysis. In the last section is a discussion of the research gaps that are being addressed in this thesis.

Table 2.1: A glance of various information management and systems integration approaches and research gaps

Approaches (Literature)	Description	Research Gap
Product Data Management (PDM) (Sung et al. 2007), (You et al. 2007), (Penoyer, et al. 2000), and others	Manages product information allowing multiple designers to work on a shared repository of design information.	Preserves only file-level dependencies between information from multiple domains. Does not capture fine-grained information dependencies such as at a parameter level.
Exchange Data Format (Pratt et al. 2005), (Fenves et al. 2005), (Fedei et al. 2005), and others	Supports information exchange between CAD/CAE/PDM systems.	Too many existing standards. Some outdated standards do not get replaced while new standards do not immediately get accepted by the industry.
NIST Core Product Model (CPM) (Fenves et al. 2005), (Sudarsan et al. 2005), (Biswas et al. 2008)	Serves as information repository for products and supports material properties information.	Needs to develop interfaces between the model and PLM systems. Needs to identify or develop standards for information exchange.
Multi- Representation Architecture (MRA) (Peak et al. 1998), (Mocko et al. 2004)	Supports CAD – CAE integration through the usage of four models each supporting different levels of product information.	The focus in MRA is on integrating geometric information and analysis information. It does not address the information link between mechanical and electrical CAD.
Constraint Linking Bridge (Collibri) (Kleiner et al. 2003)	Share data across design teams of different domains through constraints and parametric relations in CAx.	Doesn't provide information exchange between mechanical CAD and electrical CAD.

2.1 Product Data Management

The amount of engineering related information going into the design of mechatronic systems tends to be enormous. Aspects to be considered include geometric shapes of mechanical components, the electrical wiring information, information about the input and output pins of electronic circuit boards, and so on. In order to keep track of all these product data during the design process, product data management systems (PDM) are used. PDM systems manage information about a product from design to manufacture, and to end-user support. In terms of capabilities, PDM systems support five basic user functions (Liu et al. 2001):

1. Data vault and document management which provides storage and retrieval of product information;
2. Workflow and process management which controls procedures for handling product data;
3. Product structure management which handles bills of materials, product configurations, associated design versions, and design variations;
4. Parts management which provides information on standard components and facilitates re-use of designs;
5. Program management which provides work breakdown structures and allows coordination between processes, resource scheduling, and project tracking.

In terms of state-of-the-art technology, contemporary PDM systems have incorporated the use of Web-based technology. An example is a component-based product data management system (CPDM) developed by Sung and Park (Sung et al. 2007). Their CPDM system consists of three tiers: the first tier is focused on allowing

users to access the system through a Web browser; the second tier is the business logic tier that handles the core PDM functionality; and the third tier is composed of a database and vault for the physical files. This CPDM system has been implemented on the internet for a local military company that manufactures various mechatronic systems such as power cars, motorized cars, and sensitive electrical equipments.

Web-based PDM systems are also used for similarity search tasks in order to identify existing designs or components of specific shape or manufacturing-related information that may be useful for new designs or design alternatives.

You and Chen (You et al. 2007) proposed an algorithm that runs in Web-based PDM systems. In their algorithm, a target part is given with characteristic attributes, and similar parts in the database are identified based on their shape or manufacturing features. The results are sorted in the order of similarity.

There are several advantages in utilizing Web-based PDM systems. One advantage is user-friendliness: the browsers used in the PDM are the same ones used within the World Wide Web, and hence Web-based PDM systems require little amount of training. Another advantage is the great accessibility since these browsers run on various types of computers, from UNIX machines to PCs and Macintoshes. However, there are several drawbacks as well: first, the information transferring speed is limited compared to the speed of LAN or WAN; second, mistakes relating to acquiring or transferring data can occur if the system is not utilized correctly; and finally, there are major concerns regarding company's security and exposing trade secrets during the process of information transfer.

2.2 Formats for Standardized Data Exchange

PDM systems are tools that allow designers to manage and keep track of the product data throughout the entire design process. However, in order to ensure proper product configuration control, PDM systems must be able to communicate with the CAD systems that the designers use during the design process. In the context of integrated design of mechatronic products, this means communication between CAD/CAE systems of different engineering disciplines, i.e., MCAD and ECAD.

For instance, a MCAD model typically contains the following information (Pratt et al. 2005): features, which are high-level geometric constructs used during the design process to create shape configurations in the CAD models; parameters, which are values of quantities in the CAD model, such as dimensions; constraints, which are relationships between geometric elements in the CAD models, such as parallelism, tangency, and symmetry. A MCAD system cannot simply transfer such information to a PDM system or other CAD/CAE system because these systems have significantly different software architectures and data models. One potential approach towards achieving communication between various CAD, CAE, and PDM systems is through the utilization of neutral file formats, such as, for example, Initial Graphics Exchange Specification (IGES) or the Standard for the Exchange of Product Model Data (STEP). The followings are brief discussions of major data exchange format standards.

2.2.1 Initial Graphics Exchange Specification (IGES)

IGES was created for CAD-CAD information exchange. The fundamental role of IGES was to convert two dimensional drawing data and three dimensional shape data into a fixed file format in electronic form and pass the data to other CAD systems (Fenves et

al. 2005). IGES data is organized as geometric and non-geometric entities, and represented an application independent ASCII clear-text format, to and from which the native representation of a specific CAD/CAM system are mapped (Wu 2003). Major limitations of IGES include large file size, long processing time, and most importantly, the restriction of information exchange to shape data only (Fenves et al. 2005). Despite these limitations, IGES is still supported by most CAD systems and widely used for CAD information exchange.

2.2.2 ISO 10303 (STEP)

Another important neutral file format for representation of product information is STEP, also known as ISO 10303. It is an international standard for product data representation. It provides a system-independent format for the transmission of data, in computer-interpretable form, between different CAD systems or between CAD and other computer-based engineering systems. STEP is intended to cover a wide range of application areas, including aerospace, architecture, automotive, and electronic and electro-mechanical design. It also covers the various product life-cycle stages such as design analysis, process planning, manufacturing and inspection (Pratt et al. 2005).

STEP consists of several layers (Fenves et al. 2005), the top layer being a set of application protocols (APs) which address specific product classes and life-cycle stages. These APs specify the actual information exchange and are constructed from a set of modules at lower layers, called integrated resources, which are common for all disciplines. The APs relevant to electromechanical systems integration are AP-203, AP-210, AP-212, and AP-214. AP-203 protocol defines the information exchange of geometric entities and configuration control of products. This protocol captures common

modern MCAD representations including 2D drawing, 3D wireframes, surface models, and solid models, as well as information regarding assemblies, bill of materials, and design material.

Although AP-203 does not support the description of electrical characteristics, it is within the scope of AP-210 to address this need of information exchange of electrical product data between MCAD and ECAD systems. AP-210 has been developed to support the data representation and exchange of product data fro electronic assembly, interconnection, and packaging design. The scope of AP-210 is quite broad: It spans from the definition of devices and parts, to that of printed circuit boards (PCBs) and assemblies (PCAs); from the device functionality and port definition, to the assembly functional network design decomposition; from the functional requirements, to the physical implementations; and from analysis models, to manufacturing planning data (Pratt et al. 2005). On the ECAD side, AP-210 has the capability to support all the data that is described using Gerber, IDF, and other file formats; while on the MCAD side, it takes AP-203 as a subset to support the sharing of the 3D part and assembly data (Pratt et al. 2005). Other important AP relevant to CAD-CAD information exchange includes AP-212, which is concerned with electro-mechanical design and installation, and AP-214, which contains core data for automotive mechanical design process.

In addition to STEP APs that supports CAD – CAD information exchange, there are also different APs that support different type of information exchanges. AP-209 supports CAD – CAE and CAE – CAE information exchange, it provides a neutral data format representation of analysis models needed for conducting engineering analyses using heterogeneous analysis tools (Fenves et al. 2005). There is also AP203/214 PDM

Schema that supports PDM – PDM information exchange; it provides a reference information model for the exchange of a central, common subset of the data being managed within PDM system (Fenves et al. 2005). Finally, there is AP-239 entitled "Product Life Cycle Support", which supports CAD – PDM information exchange; it covers the entire history of a product from conceptual design to disposal and provides a basis for all information exchanges involved in the full CAPR (computer aided product realization) process (Pratt 2005).

Currently, there is ongoing effort in making STEP information models available in a universal format for business application developers. Lubell et al. (Lubell et al. 2004) presented a roadmap for possible future integration of STEP models with widely accepted and supported standard software modeling languages such as UML and XML. STEP provides standardized and rigorously-defined technical concepts and hence shows greater quality than other data exchange standards, but the traditional description and implementation method for STEP has failed to achieve the popularity of XML and UML (Lubell et al. 2004). Thus, emerging XML and UML-based STEP implementation technology shows promise for better information exchange ability.

2.2.3 Drawing Exchange Format (DXF)

AutoCAD DXF is another format for data exchange. It is developed by Autodesk for enabling data interoperability between AutoCAD and other programs. A DXF file is composed of pairs of codes and associated values. The codes are known as group codes and they indicate the type of value that follows. Using these group code and value pairs, a DXF file is organized into sections composed of records, which are composed of a group

code and a data item. The overall organization of a DXF file is as follows (DXF Reference 2008):

- HEADER section: contains general information about the drawing.
- CLASSES section: holds the information for application-defined classes, whose instances appear in the BLOCK, ENTITIES, and OBJECTS sections of the database.
- TABLES section: contains definition for different symbol tables, such as dimension style table, text style table, linetype table, etc.
- BLOCKS section: contains block definition and drawing entities that make up each block reference in the drawing.
- ENTITIES section: contains the graphical objects (entities) in the drawing.
- OBJECTS section: contains the non-graphical objects in the drawing.
- THUMBNAILIMAGE section: contains the preview image data.

2.2.4 Electronic-related information exchange (IDF, EDIF)

During the Electronic Computer-Aided Development of a product, the primary task of design engineers is to use CAD tools to create a detailed electronic description that contains information to support several aspects of its life cycle. One of the areas in electronic design that turns out to be a particularly big challenge is that of Printed Wiring Board Assembly (PWBA) design (Ahn et al. 2004). At this stage in the design process, Electronic CAD/CAE and Mechanical CAD/CAE tools should come together to support and complement each other in order to maximize benefits.

From the mechanical side, the PWB layout is usually determined based on mechanical and space constraints imposed by the amount of area required to physically locate all the components on the board. A desired level of integration involves being able to directly transfer PWB outlines from an MCAD tool to be used as the starting point of the design to an Electronic CAD layout tool (Ahn et al. 2004). In addition, the following information needs to be specified and transferred from MCAD to Electronic CAD: keep-in and keep-out areas on the board, height restrictions for components placement, specific features such as holes, cutouts, etc., pre-placement of components such as ICs, connectors, switches, and other devices that are linked to mechanical constraints. Even though many MCAD programs are able to generate output files that are understood by CAD layout tools, in many cases this interface uses standard formatted files that usually do not contain the information listed above and only provide basic board outline as a set of lines that the electronic and mechanical designers has to interpret in later stages of design process.

One information exchange formats for transferring electronic-related product data is the IDF (Intermediate Data Format), which specifies most of the information listed above. IDF has become the standard data exchange format for transferring PWBA data between electronic design and mechanical design (Ahn et al. 2004). The procedure for transferring PWBA data between electronic design and mechanical design during the design process is as follows (Ahn et al. 2004) (also see Figure 2.1):

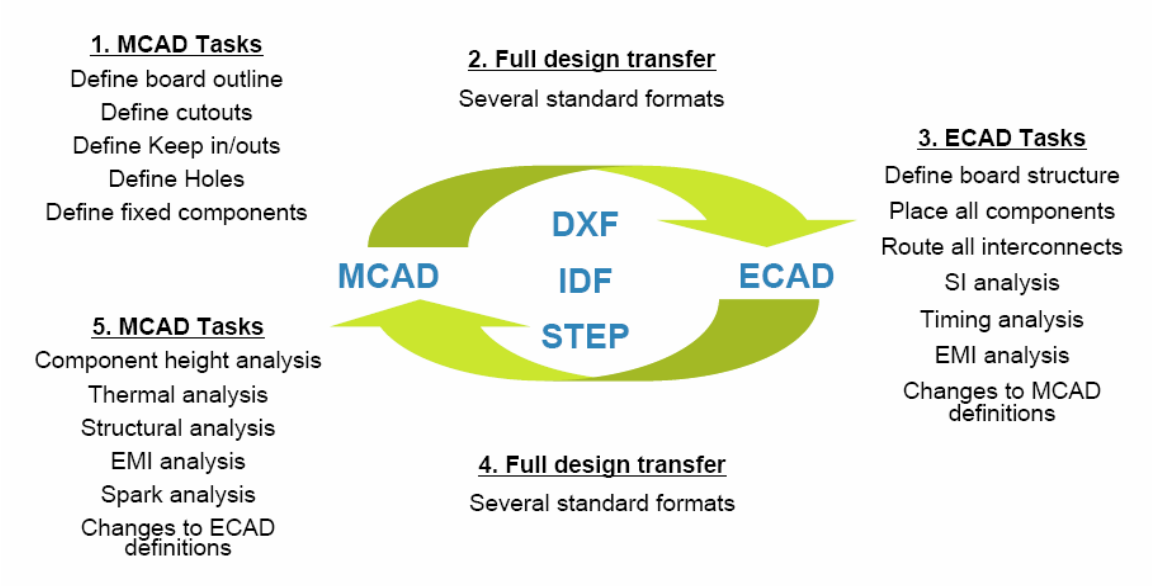


Figure 2.1: The procedures in MCAD – Electronic CAD collaboration (Hawkinson 2006)

1. In MCAD, define PWB outlines; define keep-in/keep-out areas, holes, cutouts, etc.; pre-place ICs, connectors, switches, and other fixed components.

2. Convert the above MCAD information to IDF (or other standard format such as DXF and STEP) and transfer the file to Electronic CAD.

3. In Electronic CAD, read the IDF file, write Electronic CAD model where the board structure is defined, all components are placed and all interconnect are routed. This will create an Electronic CAD file.

4. Convert the above Electronic CAD file to IDF (or other standard format) and transfer the file to MCAD.

5. In MCAD, the PWB is imported as an assembly file that has component information, including location and properties. Based on the information imported, perform component height analysis, thermal analysis, structural analysis, etc. using MCAE tools.

It is very important for the MCAD to have a fully populated library of MCAE components containing thermal and mechanical models of all parts present in the electrical layout.

Another format used for information exchange of electronic-related product data is the Electronic Design Interchange Format (EDIF). EDIF is a format used to exchange design data between CAD systems and Printed Circuit fabrication and assembly. The "Electronic" refers to the design data for electronic systems (Electronic Industries Alliance 2005). The general format of EDIF involves using parentheses to delimit data definitions. EDIF version 2.0.0 uses keywords, strings, integer numbers, and symbolic constants and identifiers. EDIF versions 3.0.0 and 4.0.0 dropped the symbolic constants entirely, using keywords instead.

One main challenge still exists in that some data within the file being outputted by MCAD is not being recognized by Electronic CAD, and vice versa. In order to resolve such a challenge, a product's packaging must take into account the physical aspects of the internal electronics while in turn, the electronics assembly – in practice the board design – must allow for the physical style and functionality of the package design. Efficiently bridging the gap between the mechanical and electronic design processes is therefore the key towards collaborative and successful product development (Evans 2007). Rather than simply passing raw dimensioning and positional data from the ECAD to MCAD environment, it would be more beneficial for the design tools to allow a bidirectional flow of comprehensive data between the two CAD environments. (Evans 2007) In other words, the ECAD must possesses the ability to import and seamless integrate 3D component data from an MCAD environment, and then pass a full and

accurate 3D representation of the board assembly back to the MCAD. To harness this potential, the electronics design system must support 3D modeling at the component level. This ability plus facilities to export accurate 3D design data would support the necessary interaction between the mechanical and electronic environments.

This section provides an overview of current state-of-the-art techniques on information exchange between mechanical and electronic domains. However, this type of information exchange (i.e., MCAD – Electronic CAD integration) is not within the scope of this thesis research, hence there is no further discussion on this topic.

2.2.5 The CAEX data format

Fedai et al. presented a data format for vendor-independent data exchange between plant engineering tools. This format is called Computer Aided Engineering Exchange (CAEX). In the classical data modeling, the necessary information about an object is specified concretely and defined in its structure (Fedai et al. 2005). In contrast to classical data modeling, the CAEX approach pursues a meta-model concept. This meta-model abstracts from concrete data structures, it no longer describes a concrete plant element, but an abstract element with as many attributes as desired. XML is used as a description language to define CAEX meta-model elements, allowing the exchange of model data through the network. Table 2.2 summarizes the various data exchange formats.

Table 2.2: Summary of various data exchange formats

ISO 10303 (STEP- Standard for Exchange of Product model data **Covers Multiple Disciplines** Description: provides a system-independent format for the transmission of data, in computer-interpretable form, between different CAD systems or between CAD and other computer-based engineering systems. Design Application Protocols: (Mechanical) AP 201- simple 2D drawing geometry. No association, no assembly hierarchy. AP 203- 3D designs of mechanical parts and assemblies AP 204- mechanical design using boundary representation AP 207- sheet metal die planning and design AP 214- core data for automotive mechanical design processes (Building) AP 202- 2D/3D drawing with association but no product structure AP 225- building elements using explicit shape representation (Electrical/Electronic) AP 210- electronic assembly, interconnect and packaging design AP 212- electro-technical design and installation AP 227- plant spatial configuration (Others) AP 209- composite and metallic structural analysis and related design AP 221- functional data and their schematic representation for process plant AP 239- product lifecycle support
IGES: Initial Graphics Exchange Specification **(CAD – CAD Exchange)** Description: convert two dimensional drawing data and three dimensional shape data into a fixed file format in electronic form and pass the data to other CAD systems
DXF: Drawing Exchange Format **(MCAD – other CAD Exchange)** Description: developed by Autodesk for enabling data interoperability between AutoCAD and other programs
IDF: Intermediate Data Format **(Electronic CAD – other CAD Exchange)** EDIF: Electronic Design Interchange Format Description: supports electronic CAD data exchange.

2.3 NIST Core Product Model

Most PDM systems and the exchange standards used for communication between CAD/CAE/PDM systems focus mainly on product geometry information. However, more attention is needed for developing standard representations that specify design information and product knowledge in a full range instead of solely geometry-oriented. At the US National Institute for Standards and Technology (NIST), an information modeling framework intended to address this issue of expanding the standard representations to a full range has been under development (Sudarsan et al. 2005). This conceptual product information modeling framework has the following key attributes (Sudarsan et al. 2005):

1. It is based on formal semantics and will eventually be supported by an appropriate ontology to permit automated reasoning;
2. It deals with conceptual entities such as artifacts and features and not specific artifacts such as motor, pumps, or gears;
3. It is to serve as information repository about products, and such information includes product description that are not at the time incorporated; and
4. It is intended to foster the development of applications and processes that are not feasible in less information-rich environments.

One major component in this information modeling framework is the core product model (CPM). The CPM is developed as a basis for future CAD/CAE information exchange support system (Fenves et al. 2005). CPM is composed of three main components (Fenves et al. 2005):

1. Function: models what the product is supposed to do;

2. Form: models the proposed design solution for the design problem specified by the function, usually the product's physical characteristics are modeled in terms of its geometry and material properties;

3. Behavior: models how the product implements its function in terms of the engineering principles incorporated into a behavioral or casual model.

CPM was further extended to components with continuously varying material properties (Biswas et al. 2008). The concept for modeling continuously varying material properties is that of distance fields associated with a set of material features, where values and rate of material properties are specified (Biswas et al. 2008). This extension of CPM uses UML to represent scalar-valued material properties as well as vector- and tensor-valued material properties.

2.4 Multi-Representation Architecture

Another approach that can be used to support the integrated designing of mechatronic system is the multi-representation architecture (MRA) proposed by Peak et al. It is a "*design analysis integration strategy that views CAD – CAE integration as an information-intensive mapping between design models and analysis models*" (Peak et al. 1998). In other words, the gap that exists in CAD/CAE between design models and analysis models is considered too large to be covered by a "single general integration bridge", hence MRA address the CAD – CAE integration problem by placing four information representations as "stepping stones" between design and analysis tools in CAD/CAE domains. The four information representations are: solution method models, analysis building blocks, product models, and product model-based analysis models.

Solution method models represent analysis models in low-level, solution method-specific form. They combine solution tool inputs, outputs, and control into a single information entity to facilitate automated solution tool access and results retrieval. Analysis building blocks represent engineering analysis concept and is largely independent of product application and solution method. It represents analysis concept using object and constraint graph techniques and has a defined information structure with graphical views to aid development, implementation, and documentation. Product models represent detailed, design-oriented product information. A product model is considered the master description of a product which supplies information to other product lifecycle tasks. It represents design aspects of products, enables connections with design tools, and supports idealization usable in numerous analysis models. And finally, product model-based analysis models contain linkages that represent design-analysis associativity between product models and analysis building blocks.

The MRA can be used to support integrated design of mechatronic systems because it has the flexibility to support different solution tools and design tools and also accommodating analysis models from diverse disciplines. Object and constraint graph techniques used in the MRA provides modularity and semantics for clearer representation of design and analysis models. Peak et al. (Peak et al. 1998) have evaluated the MRA using printed wiring assembly solder joint fatigue as a case study. Their

2.5 Constraint Linking Bridges

Computer-aided design (CAD) systems provide parametric and feature based modeling methods and support frequent model changes. In order to define and analyze various different attributes of a product, a variety of models including design models,

kinematic models, hydraulic models, electrical models, and system models are needed. Except for geometry based data transfers there is neither exchange nor integration of data for interdisciplinary product development available. Kleiner, et al. (Kleiner et al. 2003) proposed a new approach which links product models using constraints between parameters. The integration concept is based on parametric product models, which share their properties through the utilization of constraints. In their context, a virtual product is represented by partial models from different engineering disciplines and associated constraint models.

The fundamentals for the development of neutral, parametric information structures for the integration of product models are provided by existing data models from ongoing development as well as concepts from constraint logic programming (ISO, 2001). The parametric information model that Kleiner, et al. (Kleiner et al. 2003) developed uses the Unified Modeling Language (UML). The model contains the class *Item*, which represents real or virtual objects such as parts, assemblies, and models. Every object *Item* has a version (class *ItemVersion*) and specific views (class *DesignDisciplineItem Definition*). A view is relevant for the requirements of one or more life cycle stages and application domains and collects product data of the *Item* and *ItemVersion* object. The extension of STEP product data models considers the inclusion of general product characteristics (class *Property*), attributes (class *Parameter*) and restricted relationships (class *Constraint*). The information model Kleiner, et al. (Kleiner et al. 2003) developed is based on the integration of independent CAx models using its parameters. The links between CAx models are implemented using the class *Constraint*, which can set parameters of different product models in relationship to each other. On the

one hand, a constraint restricts at least one parameter and on the other hand, a parameter may be restricted by several constraints, which are building a constraint net. Different types of constraints are implemented in subclasses in order to characterize the relationship between parameters in detail.

The constraint based parametric integration offers an alternative solution compared to unidirectional process chains or file-based data exchange procedures using neutral data formats (e.g. IGES and STEP). Model structures and properties could be imported, analyzed, and exported by linking different CAx models. Kleiner et al. (Kleiner et al. 2003) developed a Java-based software system that supports product data integration for the collaborative design of mechatronic products. The software system is developed based on the constraint-based integration concept and the described information model. The software system, called Constraint Linking Bridge (Colibri), sets up connections to CAx systems in order to analyze model structures and parameters. Different interfaces to CAD and CAE systems allow the transfer of appropriate model structures and parameters as well as parameter transformation activities. Browsers for models (Model Viewer) and constraints (Constraint Viewer) are used as graphical user interfaces to analyze, specify, check, and solve constraints. The current version of Colibri contains implemented interfaces to the CAD system Pro/ENGINEER, the simulation application system MATRIXx/SystemBUILD as well as to the multi-body simulation software system AUTOLEV. Colibri is integrated into a distributed product development environment, which is called Distributed Computing Environment/Distributed File System (DCE/DFS). This infrastructure offers platform independent security and

directory services for users and groups, policy and access control management as well as secure file services storing objects in electronic vaults.

This software Colibri, though is just a prototype, it offers an alternative technology for sharing product data and illustrates the possibility of integrating difference CAx models by linking them and using constraints to specify their product development relationships. This work done by Kleiner et al. opens up opportunities for further development for supporting multidisciplinary modeling and simulation of mechatronic systems and offers user functions for data sharing.

2.6 Active Semantic Networks

In the area of electrical engineering CAD, Schaefer et al. (Schaefer et al. 1999) developed a shared knowledge base for interdisciplinary parametric product data models in CAD. This approach is based on a so-called Active Semantic Network (ASN). The ASN referred to in their approach is a shared Data Base Management System (DBMS) whose mechanisms have been adapted to the requirements of an active, data-driven design process. Such an ASN can serve as a backbone to MCAD – ECAD integration since it can be used as a common workspace for all persons involved in product design. In particular, constraints are used to model dependencies between interdisciplinary product models and cooperation, and rules using these constraints are created to help designers to collaborate and to integrate their results to a common solution. With this cooperative design approach, designers have the ability to visualize the consequences of design decisions. This visualization allows designers to integrate their design results and to improve the efficiency of the overall design process.

A semantic network or net is a graphic notation for representing knowledge in

patterns of interconnected nodes and arcs. It is a graph that consists of vertices, which represent concepts, and arcs, which represent relations between concepts (such as MCAD/ECAD constraints). An Active Semantic Network is considered as an active, distributed, and object-oriented DBMS (Roller et al. 2002). A database management system (DBMS) is computer software designed for the purpose of managing database. It is a complex set of software programs that controls the organization, storage and retrieval of data in a database. An active DBMS is a DBMS that allows users to specify actions to be taken automatically when certain conditions arise.

Constraints defined in product models are specified by rules. When a constraint is violated, possible actions include extensive inferences on the product data and notifications to the responsible designers to inform them about the violated constraints. There exists constraint propagation within the ASN. Constraints are mainly used in CAD to model dependencies between geometric objects. In the ASN, constraints are used to model any kind of dependency between product data (Roller et al. 2002).

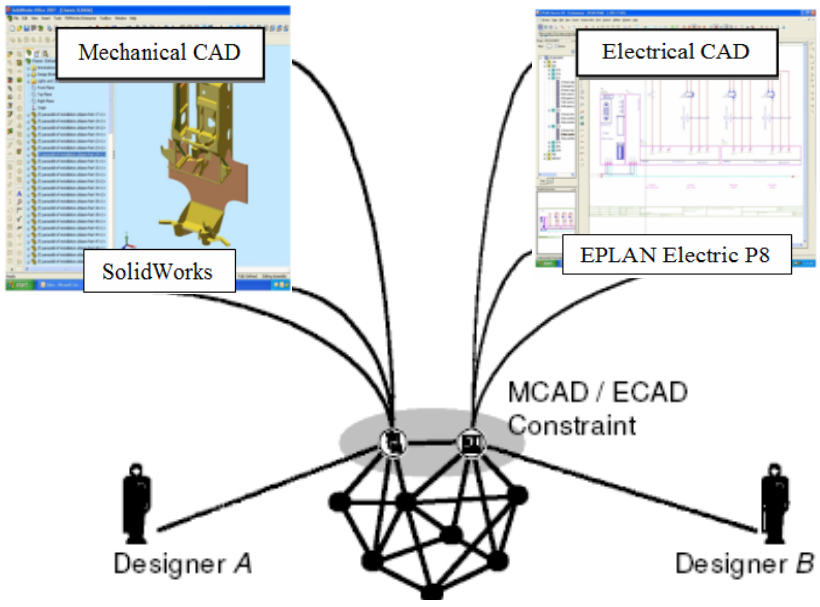


Figure 2.2 Interdisciplinary constraints between MCAD and ECAD

In the development of interdisciplinary products, constraints represent relationships between objects that depend on each other. These dependencies build the basis of the flow of information between designers in an ASN. In particular, constraints that model relationships between objects of different areas of expertise help designers understanding the interdisciplinary aspects of the product. Figure 2.2 illustrates an interdisciplinary constraint between a MCAD and an ECAD model. In the figure, the emphasis is placed on two nodes, one representing a MCAD model and the other one representing an ECAD model. The other nodes represent additional product data such as the product structure, bill of materials, cable plans, product requirements and specifications, and several other product documents. Information about the designers is also represented in the database (designer A and designer B). The dependency between the MCAD and ECAD models is indicated by the grey circle around both nodes.

As mentioned in the beginning of the section, ASN is an active DBMS that allows users to specify actions to be taken automatically when certain conditions arise. Actions initiated by an active DBMS are specified by so-called Event–Condition–Action (ECA) rules (Roller et al. 2002). An ECA rule consists of an event (e.g. a write operation on a database object) that triggers the rule. When the triggering event occurs, the condition is evaluated. The action of a rule is executed when the event is triggered and the condition is satisfied. With ECA rules, a designer can define geometrical constraints in parametric product models as well as complex and interdisciplinary inter-object dependencies.

A database object in an ASN consists of the data itself, a set of associated rules, and cooperative locks. This means that ECA rules are connected to database objects to specify their active behavior. Constraints are modeled as normal database objects that are

also subject of user modifications. ECA rules linked to a constraint object specify the active behavior of the constraint and are evaluated at run-time. The ASN uses a rule-based evaluation method for constraint propagation.

According to ECA rules, a constraint object consists of three components: an event, a condition, and an action. These components have to be specified for each constraint by users. Figure 2.3 shows a part of an object model in the ASN that contains dependencies between a MCAD and an ECAD model. Database objects are shown as circles. Links to the responsible users are inserted automatically by the system for each database object, even for constraint objects.

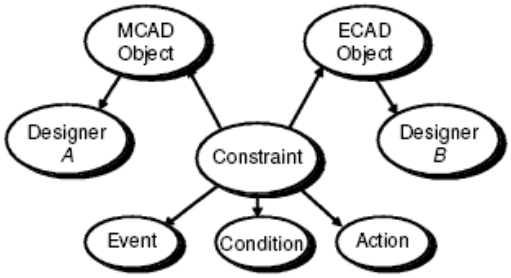


Figure 2.3 Illustration of ECA rules (Roller et al. 2002)

Often conventional databases are not sufficient for adequate representation of product and process knowledge (Bullinger et al. 2000). On the one hand, the dynamic of the development process is not being considered sufficiently, on the other hand, there is possibility to not being able to assess the consequences of one's definition. Representing product and process knowledge in the form of an ASN can provide solution to cope with the fore-mentioned problems (Bullinger et al. 2000). The objects in an ASN are not passive, they react automatically to modifications. This fact provides the possibility of an active and automatic distribution of modifications throughout the whole network.

2.7 Other Techniques

There are other techniques that are not directly related to integration of CAD and/or CAE systems, but their impacts may still contribute to CAD/CAE integration. These techniques will be briefly discussed in the following sections:

2.7.1 Knowledge Based Engineering

Knowledge based engineering (KBE) systems can be considered as one of the following system descriptions (Penoyer et al. 2000):

- A computer system used for engineering (as opposed to insurance underwriting, medical diagnostics, credit approval, or a host of other domains where computer systems or intelligent systems in particular, have been applied).
- A system focused on a distinct representation of knowledge, and the application of that knowledge to specific problem cases, where system control is not only separate from the knowledge representation, but is also independent of the exact content of the knowledge representation.
- A system characterized by deep penetration into the problem domain, capable of dealing with the details of individual problem cases, not merely general issues common to all problem cases.
- A system that reasons through a problem solving process using pattern matching and rules of logic, rather than computing a solution using a mathematical model.

Two major categories of KBE systems are distinguished by the knowledge representations they employ (Penoyer et al. 2000). The two categories are “human-interpretable” and “computer-interpretable” representations. Human-interpretable

knowledge representations include web-based repositories of knowledge, lessons learned, best practices and the like, and they are usually stored as text or HTML files. The actual application of knowledge stored in this form is mediated by a human user; the representation may be computer compatible, but the knowledge itself is not.

Computer-interpretable knowledge representations are those where the knowledge is intended to be applied to problem instances by the computer, under software control. A human user usually exercises some element of control over the process, but the knowledge representation is in some sense executable by the computer. A rule-based expert system is a prime example of such a knowledge representation. Rules are applied to problem instances by a software inference engine. The user prepares the problem cases, manages system execution, and interprets the results, but the computer-interpretable knowledge is applied to the problem instance by the computer.

The use of KBE systems can be useful in processing information flow between mechanical and electrical domains because they have the capability of storing non-geometric information. Geometric-oriented KBE systems create or synthesize detailed geometry from specifications, rules, predefined constraining geometry and user inputs. Non-geometric KBE systems are capable of handling qualitative evaluation of early design concepts; feature-based cost modeling, process planning, and process scheduling; formal logic structural modeling for validation and verification; schematic functional modeling; life-cycle knowledge processing such as wear, maintenance, disposal, etc; and other non-geometry related capabilities. Given this, it is a possibility that non-geometric KBE systems are created to store logical information in electrical engineering (such as wiring schemes) and have communication established with the geometric-oriented KBE

systems that are linked to or integrated within mechanical CAD system.

Modern CAD systems provide a suite of tools capable of managing the entire product life cycle and KBE systems are used to support many aspects of the product life cycle. Penoyer et al. believe that CAD developers should not attempt to provide the required KBE functionality as part of the product. Rather, the CAD systems should be built with a fully open architecture to allow easy integration of a broad range of KBE software (Penoyer et al. 2000). However, one major concern that rises is that the information captured for the product information exchange among the companies participating in the product delivery process needs levels of security above and beyond the security provided by the existing KBE systems for securing the company-specific proprietary information expressed in the knowledge base (Fenves et al. 2005).

2.7.2 MCAD Model Repairing Using Design History

Poor quality CAD data often results in CAD users spending time fixing or rebuilding such data from scratch on the basis of paper drawings. Early studies on poor-quality MCAD models have focused on the boundary representation (B-rep) of 3D shapes (Hoffman and Joan-Arinyo 1998). They corrected errors by mathematically analyzing the topological and geometric elements of the B-Rep model. However, Yang and Han have pointed out that correcting CAD model errors with B-Rep data has three drawbacks (Yang and Han 2006): first, when the B-Rep shape is repaired by modifying the topological and geometric elements without considering the design intent, the modeler may unintentionally distort the contextual meaning or cause the geometric structure to collapse. Second, the computations of correcting CAD model errors are prolonged because the B-Rep model can represent a wide class of objects or the B-Rep model has

complex data structure, either case will lead to computations that require a large memory. Third, even when a B-Rep shape is properly repaired, there is a problem of reusing the shape or transferring it to the upstream design team that created the CAD model because B-Rep models have no parameter or constraint information such as the engineering data on how it is created.

Yang and Han proposed an approach to repair MCAD model errors based on the design history. There are four parts to their approach (Yang and Han 2006): the first part defines the design history schema that represents the design history extracted from a CAD model. The design history refers to the chronological order in which features were created. Feature-based parametric CAD systems have a design history comprised of modeling functions, and each of the modeling functions is attached through its parameters to topological entities defined in a previous resultant shape.

The second part introduces a topological naming history fro assigning names to topological entities and for matching entities of the previous model and entities of the repaired model. There is a global matching method and a local matching method for the name-matching problem (Marcheix and Pierra 2002). The local matching method compares all the topological entities of the new model with the selected topological entity of the old model. In contrast, the global matching method saves the history of the topological evolution of both the old and new models and compares the evolution history of the old model with that of the new model.

The third part analyzes the interdependency and parametric data of feature commands. First the interdependencies of the feature commands are analyzed and relations between the corresponding feature command patterns are defined. Next the

matching relations are interpreted in the patterns of the upstream and downstream commands of the extracted design history. And finally examinations are made to determine how the parameters and constraints of each feature command depended on the parameters and constraints of the other feature commands.

The fourth part describes the necessary processes for reconstructing the design history. This part involves building a knowledge base and correcting the CAD models. The knowledge base described here is a set of rules and facts that can generate a new design history without errors. The design history extracted from a CAD model, as well as the interdependency of the feature commands, is represented as a rule and a fact in the knowledge base of an expert system. When correcting a CAD model error, the first step is to locate the error in the feature commands, and then the result and the design history information extracted from the model are transferred to the knowledge base. In accordance with the rules of the knowledge base, analyses regarding feature commands, feature command patterns, and parametric data are executed in the expert system. Through the reasoning of the expert system, the design history is reconstructed in the CAD system and new constraints and parameters are defined for the new design history.

2.7.3 Functional Modeling of Mechatronic Products

Since designers often are required to create an engineering solution that spans multiple engineering domains when designing a mechatronic product, the design elements must work together flawlessly, requiring adequate communication, monitoring, processing, and decision-making abilities. The modeling of information, such as signal flow, is vital for accurate and complete design models that support the design process (Nagel et al. 2008).

One approach to developing system models to support multi-disciplinary engineering projects and products is functional modeling. A functional model is a description of a product or process in terms of the elementary functions that are required to achieve its overall purpose (Stone and Wood 2000). A function is an operation by a device or artifact on a flow of material, energy, or signal passing through the device or artifact. Functional modeling offers significant capabilities for product design in several areas (Stone and Wood 2000):

- Systematic methods to model product functionality. Functionality is one of the fundamental constructs of product design. The ability to model a product's functionality independent of and prior to form is critical for products that meet customer demands.
- Creativity in concept generation. The ability to decompose a complicated task into simpler pieces is a critical step in the synthesis process (Ullman 2002). Identifying the key functionality of a product to be designed significantly increases the ability of designers to break problems down and find innovative solutions.
- Archival and reuse of design knowledge. A functional model provides a record of the abstract requirements of the embodied product. Recording the relationship between function and product components allows future designs to leverage the past knowledge of previous designs and designers.
- Product architecture definition. The functionality of a product, arranged graphically as a functional model, can guide the architecture of a product. Groups of functions operating on common flows lead to modules or platforms

on which a family of products may be designed.

Nagel et al. (Nagel et al. 2008) presented the concept of signal grammar which has been enumerated from functional models for both energy and material flows. The signal grammar consists of morphology guiding the interconnectivity of function and flow terms and syntax providing functional modeling templates. The developed grammar provides a structure and templates to aid in the manual and automatic assembly of functional models and guides the assembly of sub-functions, utilizing nodes to clearly establish the location of system boundaries and the required input and output flows. Signal morphology guides the arrangement of signal related function and flow terms through a functional model and is meant to clarify how and when signal flows are to be applied in mechatronic products.

Limitations of this work derive mainly from its inductive nature. Since all instances of signal functionality are not observed, which means there is no knowing of the completeness of the grammar set. However, Nagel et al. have been conducting a review of control systems literature and identified the basic control elements that must be represented in order to model mechatronic products. Nagel et al. also indicated that the future work of this signal grammar for guiding functional modeling of mechatronic systems will aim at the integration of grammar into a functional modeling CAD package. The software package will provide the designer with templates based on the syntax to aid in the functional model generation. Also, to validate the uniformity of functional models when a grammar is employed, case study functional models will be generated for mechatronic products with elements of automation that rely heavily on signal flows for overall functionality.

2.8 Evaluation of State-of-the-Art Technologies

Product data management systems manage product information from design to manufacture to end-user support. They ensure that "*right information in the right form is available to the right person at the right time*" (Liu et al. 2001). PDM can be used to work with electronic documents, digital files, and database records, including information such as product configurations, part definitions, CAD drawings, geometric models, images, engineering analysis models and results, among other product information. PDM is not limited to managing the design cycle alone but, according to user needs, can mange product conception, detailed design, prototyping and testing, manufacturing or fabrication, operation and maintenance. However, as Philpotts pointed out a decade ago, though PDM provide significant productivity gains when it is used by a workgroup, "a much greater impact will result when PDM becomes an enterprise-wide environment" (Philpotts 1996).

PDM technology, if can be implemented to all CAD/CAE systems, will significantly improve the design process of mechatronic systems because its infrastructure is user-friendly and has great accessibility. For example, Windchill developed by PTC has a browser-based user interface that uses standard HTML for bi-directional communication of form-based information and Java applets to deliver interactive application capabilities (Liu et al. 2001). Rolls-Royce AeroEngines designs and manufactures mid-range aircraft engines. They use PTC Windchill for their integrated product development environments, allowing them to maintain a quicker product delivery with increasing costs. Modern PDM systems, such as PTC Windchill, though possessing the capability of communicating with several MCAD systems, lack the

ability to communicate with ECAD systems and other CAD/CAE systems that may be used during the design of mechatronic systems.

Standardized data exchange formats, such as ISO 10303 (STEP), provide information exchange for parameterized feature-based models between different CAD systems. They provide communication between CAD systems through system independent file formats that are in computer-interpretable forms for data transmission. These data exchange formats cover a wide range of application areas: aerospace, architecture, automotives, electronic, electro-mechanical, process plant, ship building, and list can go on and on. For example, Lee and Jeong presented a study of using ISO 10303-based development methodology to develop a product model that consists several sub-models to electronically represent structural analysis and design information of steel bridges, thereby developing the foundations for the practical use of steel bridge information through linkage between three-dimensional shape data and product data of steel bridges (Lee and Jeong 2006).

Several countries such as Germany, Japan, and Korea have developed the drawing standards STEP-CDS, SCADEC, and KOSDIC, respectively, for the neutral drawing data exchange among various applications. These standards, however, are not naturally exchangeable due to the differences resulting from the "national flavors" representing the drawing elements defined in their standards (Kim and Lee 2005). Standard exchange methodology EXPRESS-X, which is an ISO standards for interoperability among the standards written based on EXPRESS, can be used to solve this exchange problem. In the conventional methodology, first the source and target model are conceptualized for conversion, and then the conversion programs are implemented using particular program

interfaces. These conversions are usually difficult and time-consuming processes and therefore the conventional methodology could not cope efficiently with the change of application platforms and the revisions of their models as it has depended on the particular programming applications (Kim and Lee 2005). In contrast, by using the ISO standardization, the conceptualization of the source and target model is identical with the implementation of conversion programs because the exchange models made based on EXPRESS-X are directly used for data conversion between the data models. Therefore, this methodology can cope flexibly with those situations as it is based on platform-independent STEP.

However, as with any standard-based exchange of information between dissimilar systems, it is impossible to convey certain elements defined in some particular CAD systems but having no counterparts in others (Pratt et al. 2005). Furthermore, past testing experience has shown that differences in the internal accuracy criteria of CAD systems can lead to problems of accuracy mismatch, which has caused many translation failures. Also, the provision of explicit geometric constraints adds possibilities for redundancy in shape models, and such geometric redundancy implies more possibility of accuracy mismatch. Hence, despite the ability to cover a wide range of applications areas, data exchange formats also have numerous problems to be solved.

NIST Core product model (CPM) and its extensions are abstract models with general semantics with specific semantics about a particular domain embedded within the usage of the models for that domain. CPM represents the products form, function, and behavior, as well as its physical and functional decompositions, and the relationships among these concepts. CPM is intended to capture product and design rationale,

assembly, and tolerance information from the earliest conceptual design stage to the full lifecycle, and also facilitates the semantic interpretability of CAD/CAE/CAM systems. The current model also supports material model construction, material related queries, data transfer, and model comparison. The construction process involves the definition of material features and choosing properties for distance field. Further research is needed to develop an API (application programming interface) between CPM and PLM systems and to identify or develop standards for the information interchange (Sudarsan et al. 2005).

Multi-Representation Architecture (MRA) is aimed at satisfying the needs in the links between CAE and CAD. These needs include (Peak et al. 1998):

1. Automation of ubiquitous analysis;

2. Representation of design and analysis associativity and of the relationships among models; and

3. Provision of various analysis models throughout the lifecycle of the product.

The initial focus of the MRA is on ubiquitous analyses, which are analyses that are regularly used to support the design of a product (Fulton et al. 1994). The MRA supports capturing knowledge and expertise for routine analysis through semantic-rich information models and the explicit associations between design and analysis models. While the MRA captures routine analysis and the mapping between design parameters and analysis parameters, there is still the opportunity for model abuse (Mocko et al. 2004). The MRA enables reuse of the analysis templates in product development. The behavior model creators (such as the analysts) and behavior model users (such as the designers) often do not have the same level of understanding of the model and thus limiting the

reuse of a model (Mocko et al. 2004). The gap between designers and analysts is decreased by providing engineering designers with increased knowledge and understanding about behavioral simulation. The future works regarding the MRA include (Mocko et al. 2004): further instantiation of the behavioral model repository, refinement of knowledge representation using ontology languages, and implementation to support instantiation with design parameters for execution.

The collaborative design system Colibri developed by Kleiner et al. offers a new approach for exchanging information across the disciplinary divide as compared to unidirectional process chains or file-based data exchange procedures using neutral data formats (e.g. IGES and STEP), however, it focuses on linking various CAx models and doesn't cover the information gap between mechanical CAD and electrical CAD systems.

In designing a mechatronic product, there are many situations that require the exchange of information between MCAD models and ECAD models. Modifications made on MCAD site may lead to significant design modifications to be made on ECAD site and vice versa. Obviously, there exist a huge number of constraints between a mechanical part of a mechatronic product and its electrical counterpart that have to be fulfilled to have a valid design configuration. As yet, such interdisciplinary constraints between models of different engineering design domains cannot be handled in a multidisciplinary Computer-Aided Engineering environment due to the lack of appropriate multidisciplinary data models and appropriate propagation method. The approach for integrated design of mechatronic systems proposed in the following chapters of this thesis focuses on supporting the information exchange of design modification between electrical and mechanical domains during the design process.

2.9 Research Gap Analysis

The first chapter stated that the focus of this thesis is to address the propagation of design modifications made to CAD models of mechanical and electrical domains. Although the various state-of-the-art technologies discussed in this chapter address various aspects of information management and systems integration, they do not address the issue of propagating changes of CAD models.

Consider data exchange formats, in particular, consider the widely-used data exchange formats – STEP, STEP through its various Application Protocols has the information of the entire CAD model such as shape, dimension, and other geometric data on the MCAD side, electrical characteristics and PCB layout information on the ECAD (both electrical and electronic) side, but STEP does not record the changes that have been made to the model. In essence, to perform the propagation of design changes (such as changes in CAD parameter value) one has to compare the STEP file of the original CAD model with the STEP file of the modified CAD model to identify the changes. This process appears to be highly inefficient.

In addition, STEP is not candidate for use in propagation of design modifications because it cannot effectively process parameters of CAD models. Modern MCAD systems generate feature-based product shape models with parameterization and constraints. Five years ago it was not possible to exchange parametric information of CAD models based on the version of STEP at that time (Choi et al. 2002 and Mun et al. 2003). Until recently, standards for CAD data exchange among different CAD systems were restricted to the exchange of pure shape information, they cannot transfer construction history (the procedure used to construct the shape model), parameters

(variables associated with dimensional or other values in the model), constraints (relationships between parameter values or between geometric/topological elements of the model), and features (shape configuration of the model) present in the model (Pratt et al. 2008). Three recently published parts of ISO 10303 try to address the above issue: ISO 10303-55 provides for the transfer of construction history information, ISO 10303-108 concerns the capture and transfer of parameter and constraint information as well as representation of 2D sketches, ISO 10303-111 provides representations for design features.

While efforts have been made to enhance STEP for creating foundations for the "standardized inter-system exchange of construction history shape models with parameterization and constraints" (Pratt et al. 2008), it still does not address the exchange of information regarding *changes* to CAD models. Other data exchange formats are even less favorable for addressing this issue than STEP. IGES, for example, already has the major limitation of large file size and long process time. IDF and EDIF address only electronic-related information. Hence STEP and other data exchange formats are not appropriate for use to achieve the objective stated in this thesis. Other technologies such as Core Product Model and MRA are not relevant for propagation of changes of CAD models either and hence achieving such propagation is a research gap that needs to be addressed.

2.10 Summary

In evaluating the literature and identifying possible research gaps in the literature, one challenge that was identified in information exchange between different domains is information overflow.

Neutral file formats and other information exchange techniques usually contain a vast amount of information, such as data for thermal, structural, or other kinds of engineering analysis, in additional to the fundamental information such as geometric features and electrical data. For the purpose of propagating design modifications between electrical and mechanical domains, using neutral file formats such as STEP can create a significant overhead because a major portion of the data in that neutral file is not being used.

Instead of transferring vast amount of information back and forth whenever a design modification occurs, the proposed method of constraint modeling and propagation provides a simple and efficient way of data exchange. In the proposed approach, whenever a design modification occurs, the corresponding changes that need to be followed are identified through user-defined constraint relationships. This approach eliminates the disadvantages of transferring large amount of information, occupying large amount of memory space, and the need to search through the vast amount of data to locate the target information.

The primary hypothesis states that integrated design of mechatronic systems can be realized through the integration of mechanical and electrical CAD systems. There are many approaches that can be used to achieve such integration, including the approaches discussed in this chapter. Constraint propagation is chosen as the approach to achieving MCAD – ECAD Integration because of the information overflow challenge.

CHAPTER 3
CONSTRAINTS IN MECHATRONIC SYSTEMS

This chapter addresses the first phase of this research: constraint identification and classification. In this chapter, a brief overview of different commonly used mechanical and electrical components of mechatronic systems is presented. Constraint relationships between attributes of mechanical and electrical components are identified and classified based on physical form, functions, and behavior of the mechatronic system. An illustrative example of robot arm is provided at the end of the chapter.

3.1 Mechatronic Systems

A typical mechatronic system consists of a mechanical skeleton (e.g., mechanical links connected by joints), actuators, sensors, controllers, signal conditioning devices, power sources, and computer or digital hardware and software. Various types of components provide different types of sensing, information acquisition, and transfer of motion. A servomotor, for example, has the capability of sensory feedback for accurate generation of complex motions. It has mechanical, electrical, and electronic components. The main mechanical components are the rotor and the stator. The electrical components include the circuitry for the field windings and rotor windings, and circuitry for power transmission. Electronic components include those needed for sensing (e.g., optical encoder for displacement and speed sensing and tachometer for speed sensing). The servomotor itself can be considered a mechatronic system because it has mechanical, electrical, and electronic components. The design of a system like the servomotor, however, would be on a lower level of details and is not the focus of this thesis.

This thesis focuses on a high level of mechatronic design, such as the design of a robot arm, in which the servomotor is considered as an electrical component. The servomotor would be one component in the ECAD model, and in the ECAD software the designer can choose from a variety of different motors alternatives. The goal in this thesis is to achieve successful propagation of design modification effects. For example, it is important to know, what if the servomotor in a robot arm gets replaced with another motor, how would other electrical and mechanical components be affected. Before going into the details of propagation of design modification, in this chapter a discussion to the common relationships between components that exist in a typical mechatronic system is provided. The following sections of this chapter are overviews of common components seen in a mechatronic system, followed by a discussion on classification of the relationships between components.

3.2 Mechanical Actuation Systems

The information exchange in the mechanical domain of designing mechatronic systems are usually information regarding the mechanisms in the mechanical system. Mechanisms are devices which can be considered to be motion converters in that they transform motion from one form to some other required form (Bolton 2003). The motion transform can be, for example, transform linear motion into rotational motion, or change the direction of a motion. Mechanical elements can include the use of linkages, cams, gears, rack-and-pinion, chains, belt drives, etc. The followings are description of typical mechanisms used in mechatronic systems and their important attributes.

3.2.1 Kinematic Chains

When considering the movements of a mechanism without any reference to the forces

involved, the mechanism is treated as being composed of a series of individual links (Bolton 2003). A link is the part of a mechanism that has motion relative to some other part. It is not necessarily a rigid body but it must be a resistant body that is capable of transmitting the required force with negligible deformation (Bolton 2003). The points of attachment to other links are the nodes. A link can have different number of nodes (see Figure 3.1). A joint is a connection between two or more links at their nodes and allows some motion between the connected links. Levers, cranks, connecting rods and pistons, slides, pulleys, belts and shafts are all example of links.

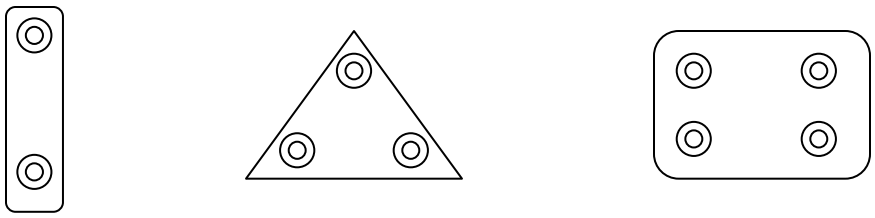


Figure 3.1: Links with two nodes, three nodes, and four nodes.

A sequence of joints and links is a kinematic chain. For a kinematic chain to transmit motion, one link must be fixed. Movement of one link will then produce predictable relative movements with the others. It is possible to obtain from one kinematic chain a number of different mechanisms by having a different link as the fixed link. The design of many mechanisms used in mechatronic systems are based on two basic forms of kinematic chains: the four-bar chain and the slider-crank chain.

The four-bar chain consists of four links connected to give four joints about which turning can occur. Figure 3.2 shows a number of forms that a four-bar chain can produce through altering the relative lengths of the links. If the sum of the length of the shortest link plus the length of the longest link is less than or equal to the sum of the lengths of the other two links, then at least one link will be capable of making a full revolution with

respect to the fixed link. If this condition is not met, then no link is capable of a complete revolution. This is known as the Grashof condition (Bolton 2003). Figure 3.2 (a) shows a double-lever mechanism, in which link 3 is fixed and the relative lengths of the links are such that no rotation can occur, links 1 and 4 can oscillate but not rotate. In Figure 3.2 (b), link 4 is shortened relative to link 1, so link 4 can rotate with link 1 oscillating; this is an example of lever-crank mechanism. In Figure 3.2 (c), both links 1 and 4 have the same length and both are able to rotate; this is an example of double-crank mechanism. By altering which link is fixed, other forms of mechanism are produced.

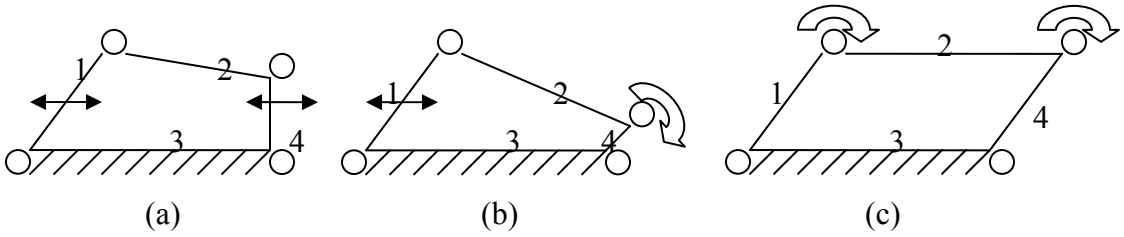


Figure 3.2: Examples of four-bar chains

Figure 3.3 shows an example of the slider-crank mechanism. In that configuration, link 3 is fixed, which means there is no relative movement between the centre of rotation of the crank and the housing in which the piston slides. Link 1 is the crank that rotates, link 2 the connecting rod and link 4 the slider which moves relative to the fixed link. When the piston (link 4) moves backwards and forwards, then the crank (link 1) is forced to rotate. Hence the mechanism transforms an input of backwards and forwards motion into rotational motion.

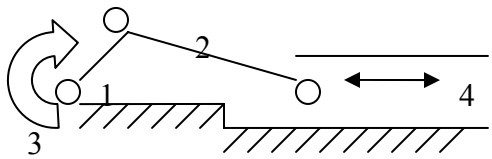


Figure 3.3: Example of a slider-crank mechanism

3.2.2 Cams

A cam is a body which rotates or oscillates and in doing so imparts a reciprocating or oscillatory motion to a second body called the follower. Figure 3.4 shows a cam and its follower. As the cam rotates, the follower raises, dwells, and falls. The lengths of times spent at each these positions depend on the shape of the cam. The rise section of the cam is the part that drives the follower upwards. Its profile determines how quickly the cam follower will be lifted. The fall section of the cam is the part that lowers the follower. Its profile determines how quickly the cam follower will fall. The dwell section of the cam is the part that allows the follower to remain at the same level for a significant period of time. The dwell section of the cam is where the profile is circular with a radius that does not change.

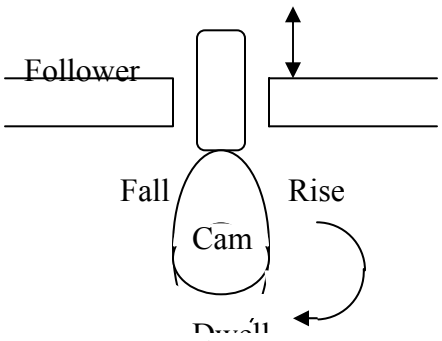


Figure 3.4: Cam and cam follower

3.2.3 Gear Trains

Gears trains (see Figure 3.5 for a simple schematic) are mechanisms that are widely used to transfer and transform rotational motion. They are used when a change in speed or torque of a rotating device is needed. An example would be a car gearbox enabling the driver to match the speed and torque requirements of the terrain with the engine power available (Bolton 2003). Rotary motion is transferred from one shaft to

another by a pair of rolling cylinders. However, there is a possibility of slip. The transfer of the motion between the two cylinders depends on the frictional forces between the two surfaces in contact. Slip can be prevented by the addition of meshing teeth to the two cylinders and the result is then a pair meshed gear wheels (Bolton 2003).

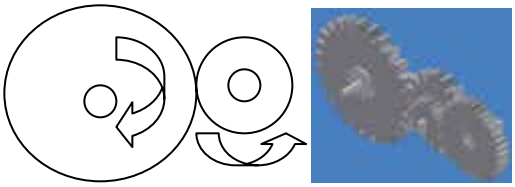


Figure 3.5: Rolling cylinders of a gear train

Gears are often used for the transmission of rotary motion between parallel shafts and for shafts that have axes inclined to one another. When two gears are in mesh, the larger gear wheel is called the spur or crown wheel and the smaller one is called the pinion. Gears for use with parallel shafts may have axial teeth with the teeth cut along axial lines parallel to the axis of the shaft; such gears are called spur gears. Alternatively the gears may have helical teeth with the teeth being cut on a helix; such gears are called helical gears. Helical gears have the advantage that there is a gradual engagement of any individual tooth and consequently there is a smoother drive and generally prolonged life of the gears (Bolton 2003). However, the inclination of the teeth to the axis of the shaft results in an axial force component on the shaft bearing. This is overcome by using double helical teeth. Figure 3.6 shows the three types of teeth cut.

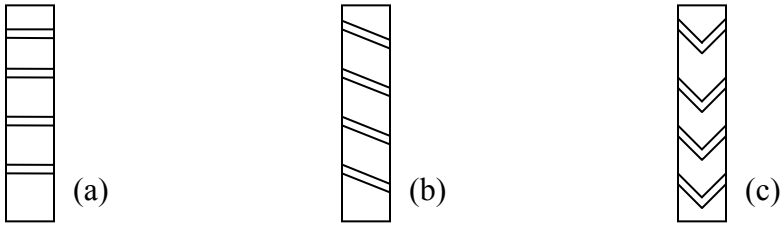


Figure 3.6: Teeth: (a) axial; (b) helical; (c) double helical

3.2.4 Ratchet and pawl

Ratchets are often used to lock a mechanism when it is holding a load. The mechanism consists of a wheel with saw-shaped teeth, called a ratchet, and an arm called pawl. The arm is pivoted and can move back and froth to engage the wheel. The shape of the teeth is such that rotation can occur in only one direction. Rotation of the ratchet wheel in a clockwise direction is prevented by the pawl can only take place when the pawl is lifted. The pawl is normally spring loaded to ensure that it automatically engages with the ratchet teeth. An example of the application of such mechanism would be: a winch used to wind up a cable on a drum uses a ratchet and pawl to prevent the cable unwinding from the drum when the handle is released.

3.2.5 Belt drives

Belt drives are a pair of rolling cylinders with the motion of one cylinder being transferred to the other by a belt. Belt drives use the friction that develops between the pulleys attached to the shafts and the belt around the arc of contact in order to transmit a torque. Since the transfer relies on frictional forces, slip can occur. The transmitted torque is due to the differences in tension that occur in the belt during operation. This differences results in a tight side and a slack slide for the belt. Figure 3.7 shows the schematic of a belt drive.

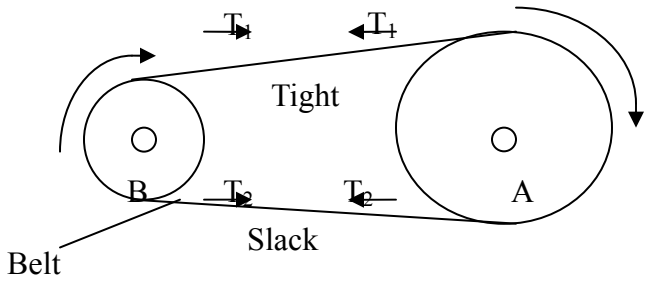


Figure 3.7: Schematic of a belt drive

If the tension on the right side is T_1 and on the slack side is T_2, as shown in Figure 3.7, then the torque on pulley A is defined by Equation 3.1 and the torque on pulley B is defined by Equation 3.2 (Bolton 2003):

$$\text{Torque on A} = (T_1 - T_2)r_A \tag{3.1}$$

$$\text{Torque on B} = (T_1 - T_2)r_B \tag{3.2}$$

where r_A is the radius of pulley A and r_B is the radius of pulley B.

Since the power transmitted is the product of the torque and the angular velocity, and since the angular velocity is v/ r_A for pulley A and v/ r_B for pulley B, where v is the belt speed, then for either pulley the transmitting power is defined by Equation 3.3 (Bolton 2003):

$$\text{Power} = (T_1 - T_2)v \tag{3.3}$$

As a method of transmitting power between two shafts, belt drives have the advantage that the length of the belt can easily be adjusted to suit a wide range of shaft-to-shaft distances and the system is automatically protected against overload because slipping occurs if the loading exceeds the maximum tension that can be sustained by frictional force (Bolton 2003). If the distance between shafts is large, a belt drive is more suitable than gears, but over small distances gears are to be preferred. However, the gear ratio is limited to about 3 because of the need to maintain an adequate arc of contact between the belt and the pulleys.

3.2.6 Bearings

When one surface rotates or slides against another surface, the resulting frictional forces generate heat that wastes energy and results in wear. The function of a bearing is to guide with minimum friction and maximum accuracy of the movement of one part moving

relative to another part. There are two main types of bearings: journal bearings and ball/roller bearings. Journal bearings are used to support rotating shafts which are loaded in a radial direction. Ball and roller bearings are used when the main load is transferred from the rotating shaft to its support by rolling contact rather than sliding contact.

3.2.7 Motor (mechanical aspects)

A motor drive system is mechanically required to rotate a shaft and its attached load. Factors that have to be considered are moments of inertia and torque. The torque required to give a load with moment of inertia I_L an angular acceleration a is $I_L\ a$. The torque required to accelerate the motor shaft is $I_M\ a_M$ and the torque required to accelerate the load is $I_L\ a_L$. The motor shaft in the absence of gearing will have the same angular acceleration and same angular velocity. The power needed to accelerate the system as a whole is the sum of these two torques, as expressed in Equation 3.4 (Bolton 2003):

$$\text{Power} = (I_M + I_L)a\omega \tag{3.4}$$

where ω is the angular velocity.

The torque to obtain a given angular acceleration will be minimized when $I_L = I_M$. Thus, for optimum performance, the moment of inertia of the load should be similar to that of the motor.

If a gear system is included in the motor allowing the motor shaft to rotate at a different angular speed to the shaft rotating the load. The gear ratio $G = \omega_L / \omega_M = a_L / a_M$, where ω_L is the angular velocity of the load, ω_M is the angular velocity of the motor, a_L is the angular acceleration of the load, and a_M is the angular acceleration of the motor. The torque and power required to accelerate the system as a whole are calculated using

Equation 3.5 and 3.6 (Bolton 2003):

$$T_M = (I_M + G^2 I_L) a_M \tag{3.5}$$

$$\text{Power} = (I_M + G^2 I_L) a_M \omega_M \tag{3.6}$$

Thus the effect of using the gearing is to give the load an effective moment of inertia of $G^2 I_L$. The torque to give a particular angular acceleration is minimized when $I_M = G^2 I_L$. Figure 3.8 shows the operating curves for a typical motor. For continuous running the stall torque value should not be exceeded. This is the maximum torque value at which overheating will not occur. As the angular speed is increased so the ability of the motor to deliver torque diminishes. Thus if higher speeds and torques are required than given by a particular motor, a more powerful needs to be selected.

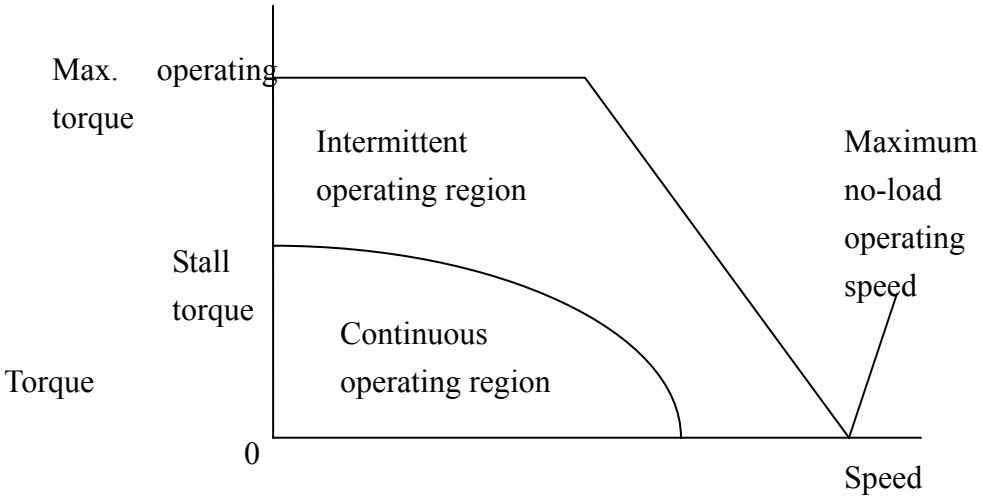


Figure 3.8: Torque-speed graph for a typical motor

3.3 Electrical Actuation Systems

The electrical actuators used in a mechatronic system include switching devices, which are used as actuators to switch on electrical systems; logic devices, such as

capacitor, inductor, and resistor; and drive systems such as DC and AC motors.

3.3.1 Switch devices

Switches are elements that are used as sensors to give inputs to systems, keyboard is an example. However, in the context as an actuator, they are used to switch on electrical devices such as electric motors or heating elements, or they are used to switch on the current to actuate devices such as solenoid valves that are used to control hydraulic or pneumatic cylinders. An electrical relay is an example of a switch device that is used in control systems as an actuator.

Relays are electrically operated switches in which changing a current in one electrical circuit switches a current on or off in another circuit. When a current is passed through the solenoid of a relay, a magnetic field is produced to attract the iron armature moving the push-rod to close the normally open (NO) switch contacts and open the normally closed (NC) switch contacts. There are also time-delay relays, which are control relays that have a delayed switching action. The time delay is usually adjustable and is initiated when a current flows through the relay coil or when it ceases to flow through the coil. Figure 3.9 shows symbols that represent switches in an ECAD system (EPLAN Electric P8).

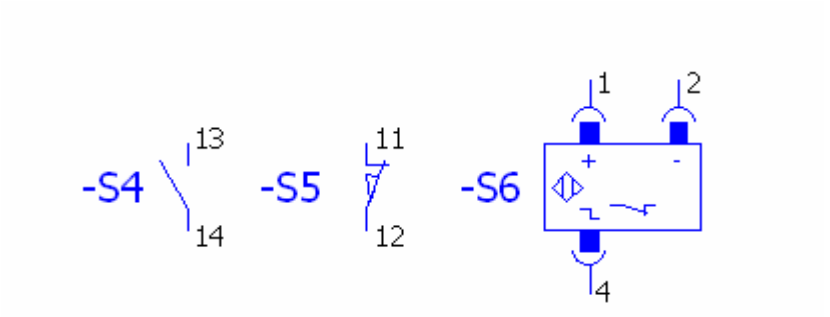


Figure 3.9: Symbols for different types of switches in EPLAN Electric P8

3.3.2 Logic devices

Capacitor

Electrical capacitance is a measure of the amount of electric charge stored for a given electric potential. The most common form of charge storage device is a two plate capacitor, which consists of two conducting surfaces separated by a layer of insulating medium called dielectric. The dielectric is an insulating medium through which an electrostatic field can pass. The main purpose of the capacitor is to store electrical energy.

The capacitance of a capacitor is defined as the amount of charge required to create a unit potential difference between its plates. When a DC voltage is applied to a capacitor, it momentarily acts like a short circuit and the capacitance acts like an open circuit. When a capacitor is connected to a source of alternating voltage, the continuous charge and periodical reversal of the applied voltage causes a continuous change in state of the capacitor with a continuously changing current. A capacitor acts like a short circuit for an AC voltage (Appuu Kuttan 2007).

Capacitance of a capacitor depends on the dielectric constant k, area of one side of the plate A, number of plates N, and separation of the plate surfaces d, and is expressed by Equation 3.7:

$$C = \frac{kA(N-1)}{d} \tag{3.7}$$

For a capacitor, voltage is a natural output variable and current is natural input variable, and the voltage across a capacitor cannot change instantaneously unless an infinite current is applied.

Inductor

A wire wound in the form of a coil makes an inductor. The property of an inductor

is that it always tries to maintain a steady flow of current and opposes any fluctuation in it. When a current flows through a conductor, it produces a magnetic field around it in a plane perpendicular to the conductor. When a conductor moves in a magnetic field, an electromagnetic force is induced in the conductor. The property of the inductor due to which it opposes any increase or decrease in current by the production of a counter emf is known as self-inductance. The emf developed is proportional to the rate of current through the inductor, and mathematically it depends on the amount of current, the voltage developed, and a proportionality constant that represents the self-inductance of the coil.

Resistor

Electrical resistance is a measure of the degree to which an object opposes the electric current through it. The electrical resistance of an object is a function of its physical geometry. The resistance is proportional to the length of the object and inversely proportional to the cross-section area of the object. All resistors possess some degree of resistance. The resistances of some resistors are indicated by numbers. This method is used for low value resistors. Most resistors are coded using color bands, and the way to decode these bands is as follows: the first band gives the resistance value of the resistor in ohms. The fourth band indicates the accuracy of the value. Red in this region indicates 2%, gold indicates 5%, and silver indicates 10% accuracy.

Figure 3.10 shows symbols that represent the different logic devices in EPLAN Electric P8.

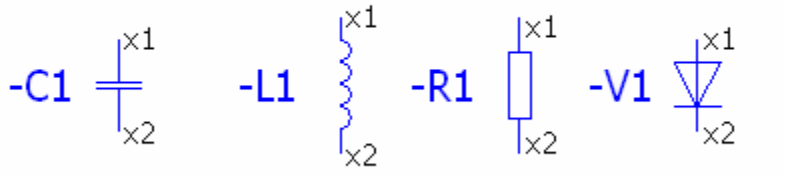


Figure 3.10: Symbols for different types of logic devices in EPLAN Electric P8

(Capacitor, Inductor, Resistor, Semiconductor)

3.3.3 Driving systems

Electric motors are frequently used as the final control element in positional or speed-control systems. Motors are typically classified into two main categories: DC motors and AC motors. DC motors with field coils are classified as series, shunt, compound, and separately exited according to how the field windings and armature windings are connected. The series wound motors have the armature and field coils in series. Such a motor exerts the highest starting torque and has the greatest no-load speed. With light loads there is a danger that a series wound motor might run at too high a speed. The shunt wound motor have the armature and field coils in parallel. It provides the lowest starting torque, a much lower no-load speed and has good speed regulation. Because of this almost constant speed regardless of load, shunt wound motors are widely used (Bolton 2003). The compound motor has two field windings, one in series with the armature and one in parallel. Compound wound motors aim to get the best features of the series and shunt wound motors: the high starting torque and the good speed regulation. The separately excited motor has separate control of the armature and field currents and can be considered to be a special case of the shunt wound motor.

Alternating current motors are typically classified into two groups: single phase and polyphase, and each group further sub-divide into induction and synchronous motors. Single phase motors tend to be used for low power requirements while polyphase motors are used for higher powers. Induction motors tend to be cheaper than synchronous motors and are thus widely used. Common types of AC motors that are used in mechatronic systems are single-phase squirrel-cage induction motor, three-phase induction motor, and three-phase synchronous motor.

3.4 Other Electrical Devices

3.4.1 Sensors

Measurements of variables are needed for monitoring and control purposes. Typical variables that need to be measured in a mechatronic system for data acquisition and controls are (Cetinkunt 2007):

1. Position, velocity, acceleration
2. Force, torque, strain, pressure
3. Temperature
4. Flow rate
5. Humidity

The measurement device is the senor. A sensor is placed in the environment where a variable is to be measured. The sensor is exposed to the effect of the measured variable. There are three basic phenomenons in effect in any sensor operations (Cetinkunt 2007):

1. The change in the measured physical variable (i.e., pressure, temperature, displacement) is translated into a change in the property (resistance, capacitance, magnetic coupling) of the sensor. This is known as the transduction. The change of the measured variable is converted to an equivalent property change in the sensor.
2. The change in the property of the sensor is translated into a low-power-level electrical signal in the form of voltage or current.
3. This low-power sensor signal is amplified, conditioned, and transmitted to an intelligent device for processing.

3.4.2 Power Supply

Power supply is the source of electrical power. It supplies electrical or other types of energy to an output load or group of loads. The term power supply covers various types of power sources:

- Conversion of one form of electrical power to another desired form and voltage, e.g., linear regulator, rectifier, and electrical inverter.
- Batteries
- Chemical fuel cells and other forms of energy storage systems.
- Solar power
- Generators

In ECAD software such as EPLAN Electric P8 the power supply that are created in the ECAD model are generators. They are low voltage, low power DC power supply units that are commonly integrated with devices such as computers and household electronics. Power supplies for electronic devices can furthermore be broadly divided into linear and switching power supplies. An AC powered linear power supply usually uses a transformer to convert the voltage from the wall outlet to a usually lower voltage. If it is used to produce DC, a rectifier is used. A capacitor is used to smooth the pulsating current from the rectifier. The linear supply is usually a relatively simple design that becomes increasingly bulky and heavy for high current devices and voltage regulation in a linear supply can result in low efficiency. A switched-mode power supply works in a different way, AC main input is directly rectified without the use of a transformer, to obtain a DC voltage. This voltage is then sliced into small pieces by a high-speed electronic switch.

3.4.3 Programmable Logic Controllers

The physical shape of all PLCs made by many different companies all has the same form: it is rack mounted with standard-size slots to plug-in I/O interface units (Centikunt 2007). A typical PLC rack starts with a power supply and a CPU module plugged into a backplane of interface bus. A rack may have different number of slots. PLCs have the following properties:

1. PLCs have modular design. If a different type of I/O signal needs to be processed, all is needed is to add a different I/O interface module and modify software.
2. PLCs have a rugged design suitable for harsh industrial environments again high temperature variations, dust and vibrations.
3. Programming of PLCs is mostly done using ladder logic diagrams.

A PLC in EPLAN Electric P8 is shown in Figure 3.11 below. In the figure it is seen that the PLC has I/O interface for power supply (24V), ground (0V), 1PE, 2PE, 1+, 1-, etc.

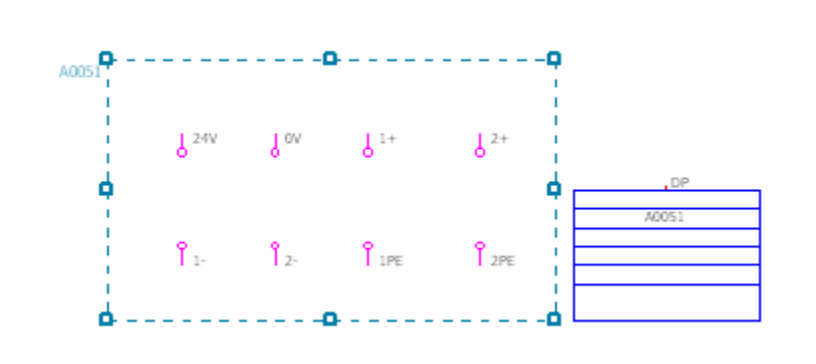


Figure 3.11: PLC in EPLAN Electric P8

3.5 Mechanical Constraints

3.5.1 Geometric Constraints

Most CAD systems allow the creation of variational models with parameterization, constraints and features. The set of common geometric constraint types is listed as follows (Klein 1998):

• Parallelism – this has an undirected form and a directed form with one reference element. There is also a dimensional subtype, in which a constrained distance is specified.

• Point-distance – in the directed case the reference element may be either point, line, or plane. Multiple points may be constrained. In the undirected case, the number of constrained points is limited to two, and a dimensional value is required.

• Radius – has a dimensionless form, for example, “the radii of all these arcs are the same”, and a dimensional form, for example, “the radii of all the constrained arcs have the same specified value”.

• Curve-length – asserts that the lengths of all members of a set of trimmed curves are equal. There is a dimensional form allowing the value of the length to be specified.

• Angle – constraints a set of lines or planes to make the same angle with a reference element, or in the undirected case specifies the angle between precisely two such elements.

• Direction – a vector-valued constraint used for constraining the directional attributes of linear elements such as lines or planes.

• Perpendicularity – there may be either one or two reference elements (lines or planes), and all the constrained elements are required to be perpendicular to them. There is also an undirected form in which two or three elements are required to be mutually perpendicular.

• Incidence – in its simplest form, it simply asserts that one or more constrained entities are contained within that reference element.

• Tangency – may be used to specify multiple tangencies between a set of reference elements, and as set of constrained elements.

• Coaxial – constrained a set of rotational elements to share the same axis or to share a specified reference axis.

• Symmetry – constrains two ordered sets of elements to be pair-wise symmetric with respect to a given line or plane.

• Fixed – used to fix points and directions in absolute terms for anchoring local coordinated systems in global space.

3.5.2 Kinematics Constraints

Kinematics is a branch of mechanics that describes the motion of objects without the consideration of the masses and forces that bring about the motion (Appuu Kuttan 2007). Kinematics studies how the position of an object changes with time. Position is measured with respect to a set of coordinates. Velocity is the rate of change of position. Acceleration is the rate of change of velocity. In designing mechatronic systems, the kinematics analysis of machine elements is very important. Kinematics determines the position, velocity, and acceleration of machine links. Kinematics analysis helps to find the impact and jerk on a machine element.

3.5.3 Force Constraints

In mechanical engineering, and in particular, in dealing with "machines in mechatronics", it often involves the study of relative motion between the various parts of a machine as well as the forces acting on them, hence the knowledge in this subject of "forces" is very essential for an engineer to design the various parts of mechatronic systems (Appuu Kuttan 2007). Force is an important factor as an agent that produces or tends to produce, destroys or tends to destroy motion. When a body does not move or tend to move, the body does not have any friction force. Whenever a body moves or tends to move tangentially with respect to the surface on which it rests, the interlocking properties of the minutely projected particles due to the surface roughness oppose the motion. This opposition force that acts in the opposite direction of the movement of the body is the force of friction. Both force and friction play an important role in mechatronic systems.

In considering the force constraint in mechanical systems, there are three major parameters that can affect the mechanical systems: the stiffness of the system, the forces opposing motion (such as frictional or damping effects), and the inertia or resistance to acceleration (Bolton 2003). The stiffness of a system is described by the relationship between the forces used to extend or compress a spring and the resulting extension or compression. The inertia or resistance to acceleration exhibits the property that the bigger the inertia (mass) the greater the force required to give it a specific acceleration.

Lubrication between journals and bearings or sliders and bearings is also an important aspect of mechanical design in mechatronic systems.

3.5.4 Energy Constraints

Energy is a scalar physical quantity that is a property of objects and systems which is conserved by nature. Energy can be converted in a variety of ways. An electric motor converts electrical into mechanical and thermal energy, a combustion engine converts chemical into mechanical and thermal energy, and so on. In physics, mechanical energy describes the potential energy and kinetic energy present in the components of a mechanical system. If the system is subject only to conservative forces, such as only to gravitational force, by the principle of conservation of mechanical energy the total mechanical energy of the system remains constant.

3.5.5 Material Constraints

The various machine parts of the mechatronic system often experience different loading conditions. If a change of motion of the rigid body (the machine parts) is prevented, the force applied will cause a deformation or change in the shape of the body. Strain is the change in dimension that takes place in the material due to an externally applied force. Linear strain is the ratio of change in length when a tensile or compressive force is applied. Shear strain is measured by the angular distortion caused by an external force. The load per unit deflection in a body is the stiffness. Deflection per unit load is the compliance. If deformation per unit load at a point on the body is different from that at the point of application of the load then compliance at that point is called cross compliance. In a machine structure cross compliance is an important parameter for stability analysis during machining.

The strength of a material is expressed as the stress required causing it to fracture. The maximum force required to break a material divided by the original cross-sectional

area at the point of fracture is the ultimate tensile strength of the material. It is obvious that the stress allowed in any component of a machine must be less than the stress that would cause permanent deformation. A safe working stress is chosen with regard to the conditions under which the material is to work. The ratio of the yield stress to allowable stress is the factor of safety.

3.5.6 Tolerance Constraints

The relationship resulting from the difference between the sizes of two features is the fit. Fits have a common basic size. They are broadly classified as clearance fit, transition fit, and interference fit. A clearance fit is one that always provides a clearance between the hole and shaft when they are assembled. A transition fit is one that provides either a clearance or interference between the hole and the shaft when they are assembled. An interference fit is one that provides interference all along between the hole and the shaft when they are assembled.

The production of a part with exact dimensions repetitively is usually very difficult. Hence, it is sufficient to produce parts with dimensions accurate within two permissible limits of size, which is the tolerance. Tolerance is provided on both sides of the basic size (bilateral tolerance) or on one side of the basic size (unilateral tolerance). The ISO systems of tolerance provides for a total of 20 standard tolerance grades.

In any engineering industry the components manufactured should also satisfy the geometrical tolerances in addition to the common dimensional tolerances. Geometrical tolerances are classified as:

- Form tolerance—straightness, flatness, circularity, cylindricity, profile of any line, and profile of any surface.

- Orientation tolerance—parallelism, perpendicularity, and angularity.
- Location tolerance—position, concentricity, co-axiality, and symmetry.
- Run-out tolerance—circular run-out and axial run-out.

3.6 Electrical Constraints

3.6.1 Motor Torque

The torque generation in any electric motor is essentially the conversion process of converting electrical energy into mechanical energy. It resulted from the interaction of two magnetic flux density vectors: one generated by the stator and one generated by the rotor. In different motor types, the way these vectors generated is different (Centikunt 2007). For instance, in a permanent brushless motor the magnetic flux vector is generated by the current in the windings. In the case of an AC induction motor, the stator magnetic flux vector is generated by the current in the stator winding, and the rotor magnetic flux vector is generated by induced voltages on the rotor conductors by the stator field and resulting current in the rotor conductors. The torque production in an electric motor is proportional to the strength of the two magnetic flux vectors (stator's and rotor's) and the sine of the angle between the two vectors.

In using ECAD software, there is a list of motors to choose from. Each motor selection provides different value of torque and sometimes requires different power supply input. In designing a mechatronic system, whether it is a robot arm or an elevator, when the mechanical features change, such as geometry (e.g., robot arm length, size of the elevator), the motor would be providing a different torque to support the modified system. And if the current motor is too weak or too strong, a replacement is needed.

3.6.2 System Controls

The control system provides a logical sequence for the operating program of the mechatronic system. It provides the theoretical values required for each program step, it continuously measures the actual position during motion, and it processes the theoretical versus actual difference (Rembold et al. 1993). In controlling a robot, for example, there are two types of control techniques: point-to-point and continuous path. The point-to-point involves the specification of the starting point and end point of the robot motion and requiring a control system to render feedbacks at those points. The continuous path control requires the robot end effector to follow a stated path from the starting point to the end point.

In creating a programmable logic controller in ECAD software, it is important to identify the right connection point. It is rare for connection points to have same designation.

3.6.3 Mounting Restriction

For an electrical component to work appropriately in a mechatronic system, it needed to be mounted onto the mechatronic system. Hence in creating an electrical component in ECAD software, there are mounting data and mounting restriction carried by that particular electrical component. For example, a programmable logic controller with part number "BECK.BK3100" has the following attributes: 1.7 N in weight, 49mm in width, 100 mm in height, and 68mm in depth. The area based on the width and height would be the space requirement for this particular electrical device. Hence this PLC with part number "BECK.BK3100" has a space requirement of 4900 mm^2. In addition, the device can have mounting clearance width, height, and depth.

3.6.4 Power Supply Specifications

In EPLAN Electric P8 the power supplies that are being created in the ECAD model are generators. Each power supply has its own specification of voltage and current. For example, power supply with part numbers “BLO.GLC230/24-1”, “BLO.GLC230/24-5”, “BLO.GLC230/24-10”, “BLO.GLC400/24-1”, “BLO.GLC400/24-5”, “BLO.GLC400 /24-10” all are DC power supply that has primary and secondary side miniature fuses LED for optical display of output voltage. However, the first three are 230V DC supply with output of 24V DC and 1A, 5A, and 10A of current, respectively; and the latter three 400V DC supply with output of 24V DC and 1A, 5A, and 10A of current, respectively.

3.6.5 Power Cable Specifications

A power cable is an assembly of two or more electrical conductor usually held together with an overall sheath. Figure 3.12 below shows the schematic of a power cable. Similar to other electrical components, EPLAN Electric P8 also provides a variety of power cable for selection. Besides the standard geometric features of cross-section diameter, sleeve diameter, and strip length, there are also specific features such as minimum bending radius for occasional non-persevering movements or other specification such as oil-resistances, use of conductor with improved behavior in case of fire, for use in industrial environment, etc. The power cable in EPLAN Electric P8 also carries material properties information such as copper weight and cable weight in N/km.

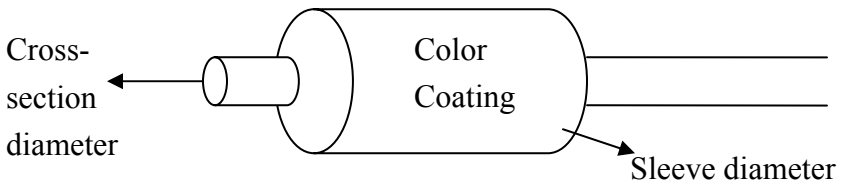


Figure 3.12 Schematic of a power cable

3.7 Constraints Classification

The study of constraints has been a well developed field for the mechanical engineering side. For example, geometric constraints, specifying the relations between elements of a geometric model, and geometric constraint solving not only has applications in Computer-Aided Design, but also in tolerance analysis and geometric theorem proving. However, in the electrical engineering world, the concepts similar to that of geometric constraints and material constraints have not been developed. In order to achieve better communication between the electrical and the mechanical domains, all the previously discussed constraint types should be further classified to later provide a way for direct mapping between the mechanical aspects and the electrical aspects of the mechatronic system. In this proposed constraints classification, the constraints are classified based on functions, forms, and behavior. The three classes of constraints are fundamentally different and each requires a specific model for modeling the characteristics. The emphasis in this thesis is on constraints based on functions.

3.7.1 Constraints based on Function

Constraints based on function are ones that relate attributes of mechatronic systems by equation:

- Geometric features that are defined by dimensions or other attributes of the geometry, e.g., area, volume, perimeter, etc.
- Kinematics constraints, in which the attributes of the components are related by time dependent functions, e.g., velocity is a function of displacement and time, acceleration is a function of velocity and time, etc.

- Force constraints, in which the dynamics aspects of the mechatronic system are expressed in a function, e.g., force is a function of mass and acceleration, torque is a function of moment arm and force, etc.
- Motor torque, is under the umbrella of force constraints, it is listed on the electrical side because it is a characteristic of an electrical component.
- Mounting area restriction involves dimensions and space requirements. This constraint involves geometric features; it is listed on the electrical side because it is a characteristic of an electrical component.

3.7.2 Constraint based on Form

Constraints based on form are ones that relates a component's physical appearance:

- Form-based geometric feature: parallelism, perpendicularity, symmetry, etc.
- Tolerance, which describes to what extent, is the form acceptable.
- Physical appearance related specifications, e.g., power cable specification.

3.7.3 Constraints based on Behavior

Constraints based on behavior are ones that relates a component's response to certain actions:

- Material constraints, such as stress and strain, which is used to measure a component's response to a load, the response can be, for example, deformation.
- System control, upon receiving signal from a controlling device, such as programmable logic controller, the component, e.g., the gripper of a robot arm, respond according to the signal to perform certain action.

3.8 Illustrative Example: Constraints in a Robot Arm

3.8.1 Overview of the Robot Arm

A robot is a mechatronic system capable to replace or assist the human operator in carrying out a variety of physical tasks. The interaction with the surrounding environment is achieved through sensors and transducers and the computer-controlled interaction systems emulate human capabilities. The case study example investigated is the SG5-UT robot arm designed by Alex Dirks of the CrustCrawler team (see Figure 3.13).

List of major mechanical component:

- Base and Wheel plates
- Links: Bicep, Forearm, Wrist
- Gripper
- Joints: shoulder, elbow, and wrist
- Hitec HS-475HB servos (base, wrist and gripper)
- Hitec HS-645MG servos (elbow bend)
- Hitec HS-805BB servos (shoulder bend)

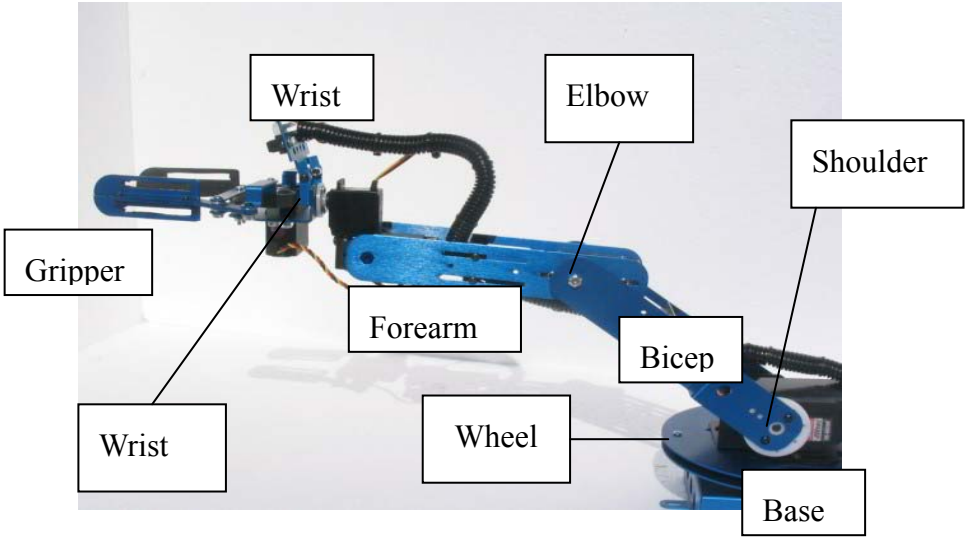
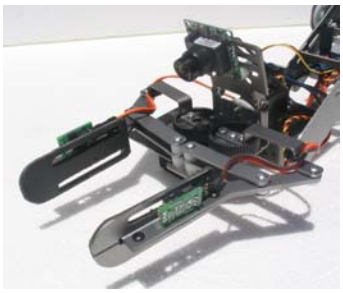
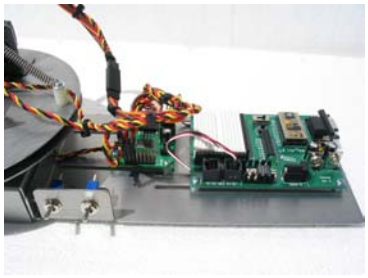


Figure 3.13: The CrustCrawler SG5-UT robot arm

Table 3.1 below show the three main elements of the CrustCrawler SG5-UT robot arm.

Table 3.1: Main components of CrustCrawler SG5-UT

Component	Description
Gripper 	The Gripper: the most critical aspect of any robot arm is in the design of the gripper. A robot arm's usefulness and functionality is directly related to the ability to sense and successfully manipulate its immediate environment. The gripper drive system consists of a resin gear train driven by HS 475 servo.
Servomotors  	The servos are needed to provide motion to the various mechanical links as well as the gripper. The mounting site of the servos and the power routing to servos and supporting electronics are some of important aspects to considered in the design of this robot arm
Microcontroller board 	The microcontroller board is essential for communication between the robot and the PC, providing users the ability to manipulate the robot. It is important to have accurate information on the pin connections and the corresponding components that are being controlled.

3.8.2 Modeling Constraints for SG5-UT

STEP 1: List all components of the SG5-UT robot arm and their attributes and classify the components in either the mechanical domain or the electrical domain. These components are listed in Figure 3.14.

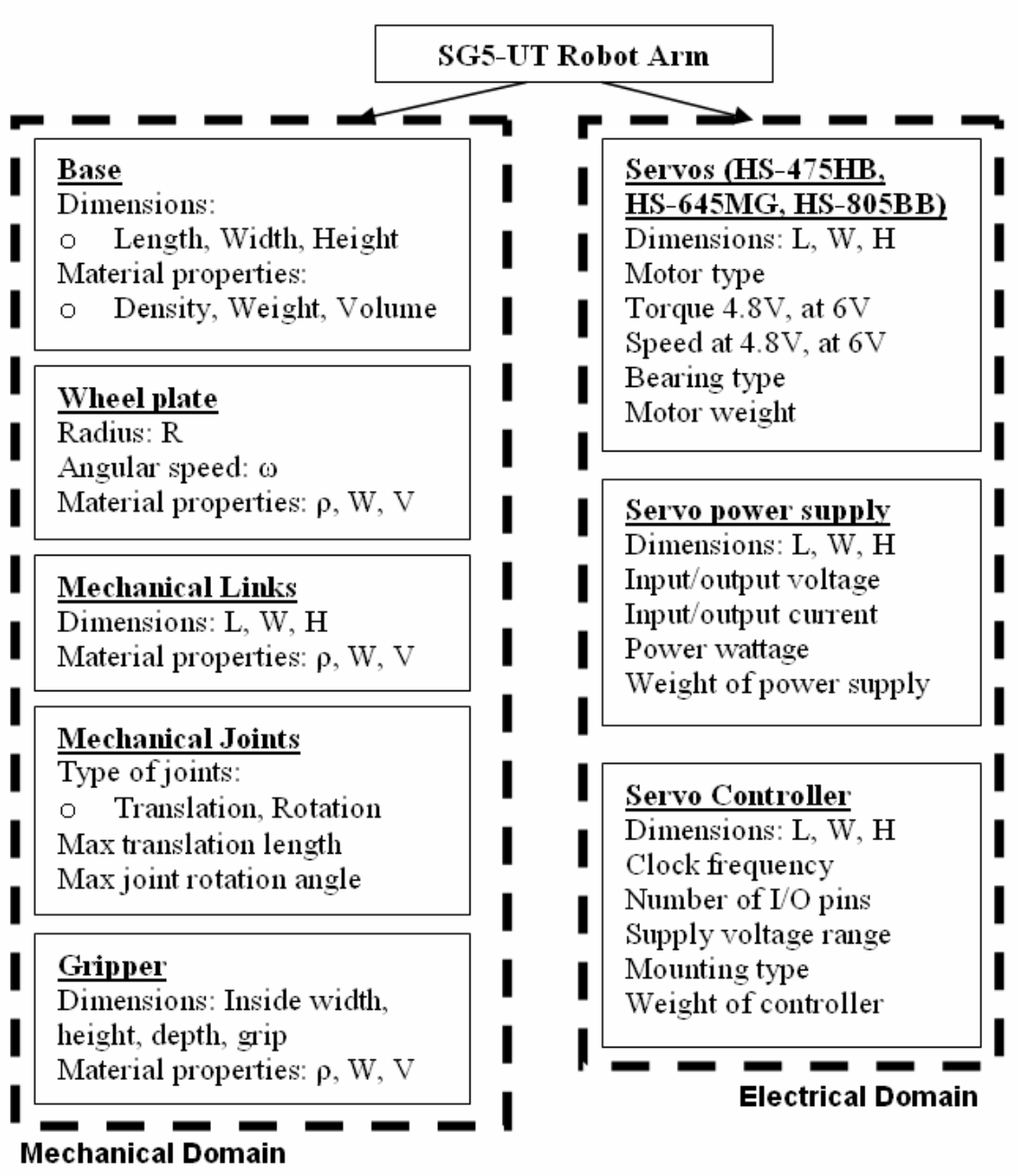


Figure 3.14: Mechanical and electrical attributes of the robot arm

STEP 2: Based on the attributes of the component, draw the constraint relationship between the components in the domain and appropriately label the constraint by the constraint categories, as presented below in Figure 3.15 and in Table 3.2 and 3.3. In Figure 3.15, Solid line represents constraints within the domain, M is mechanical and E is electrical, and dashed line represents cross-disciplinary, C.

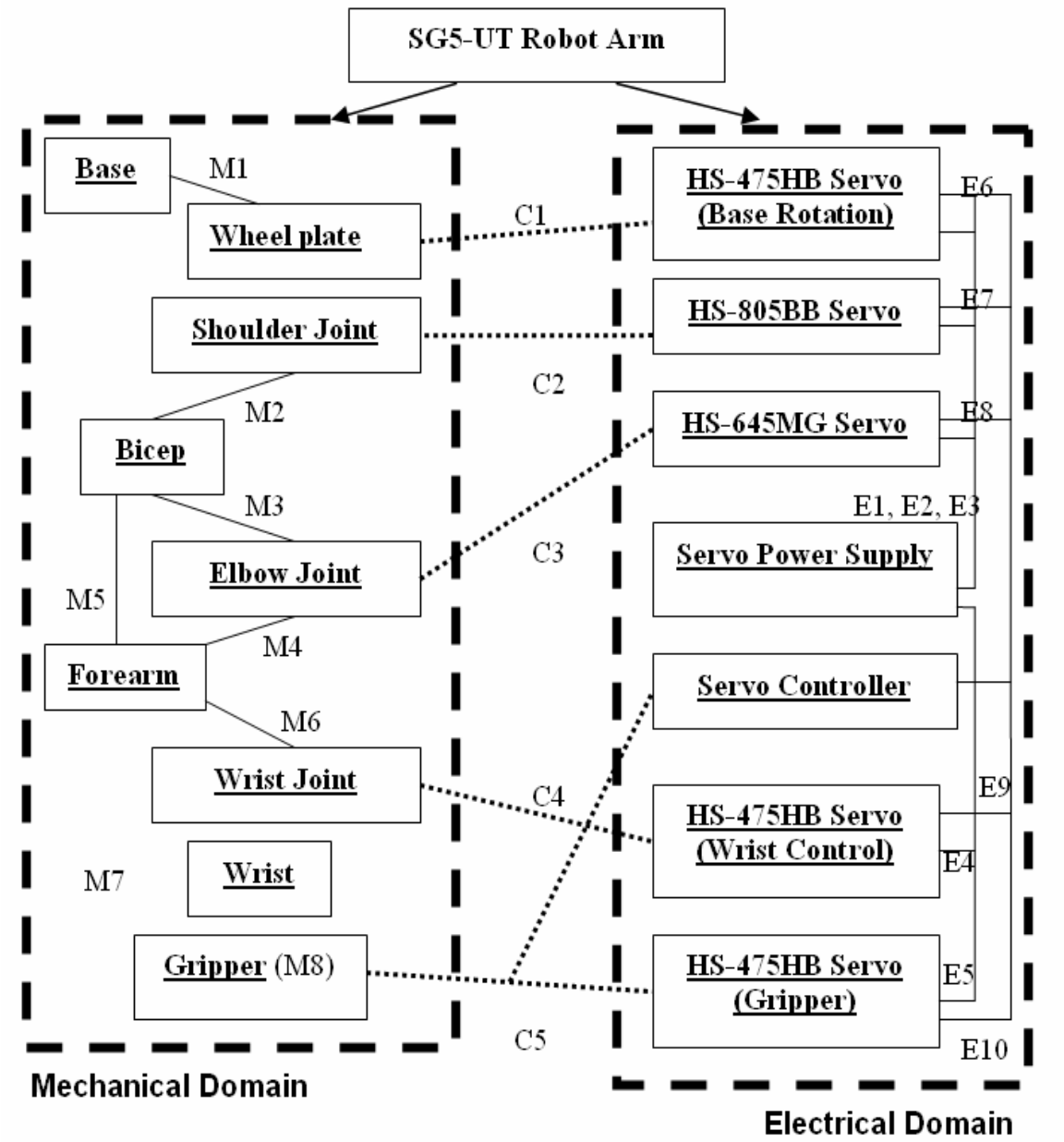


Figure 3.15: Constraint model for the robot arm

Table 3.2: Mechanical constraints in SG5-UT

	Constraint Type	Constraint Description
M1	Geometric: fixed Between the base and the wheel plate.	The coordinate of the base contact point with the wheel plate is the coordinate of the centre of the wheel plate.
M2	Geometric: coaxial Between the bicep and shoulder joint.	The axis of shoulder joint is coaxial with the axis of bicep rotation.
M3	Geometric: fixed Between the bicep and elbow joint.	The coordinate of the bicep contact point with the forearm is the coordinate of elbow joint.
M4	Geometric: coaxial Between elbow joint and forearm.	The axis of elbow joint is coaxial with the axis of forearm rotation.
M5	Geometric: angle Between bicep and forearm.	The angle between the bicep and forearm is between 90 and 270 degrees. The forearm is not permitted to crush into the bicep.
M6	Geometric: fixed Between forearm and wrist joint.	The coordinate of the wrist contact point with the forearm is the coordinate of wrist joint.
M7	Geometric: coaxial Between wrist joint and gripper.	The axis of wrist joint is coaxial with the axis of gripper rotation.
M8	Geometric: symmetry Gripper.	The left half of the gripper and the right half of the gripper are symmetric.

Table 3.3: Electrical constraints in SG5-UT

	Constraint Type	Constraint Description
E1 ~ E5	Maximum Torque Between the five servos and the power supply.	There is a maximum torque for a particular given voltage from the power supply.
E6 ~ E10	System Control Between the five servos and the servo controller.	The servo controller determines the appropriate drive signal to move the actuator towards its desired position.

STEP 3: Based on the attributes of the component, draw the constraint relationship between the components across the domains and appropriately label the constraint by the constraint categories, as presented below in Table 3.4.

Table 3.4: Cross-disciplinary constraints in SG5-UT

	Constraint Type	Constraint Description
C1	Kinematics – Force – Motor torque Between the wheel plate and base rotation servo.	The rotation speed of the wheel plate is dependent on the weight of the entire arm structure and the torque provided by the base rotation servo.
C2 C3 C4	Geometry – Force – Motor Torque Between servos and mechanical links	The torque required about each joint is the multiple of downward forces (weight) and the linkage lengths. This constraint exists in each lifting actuators.
C5	System Control – Kinematics – Force Between gripper, gripper servo, and servo controller	The servo controller determines the current state of the gripper given the current state of the actuators (position and velocity). The controller also adjusts the servo operation given the knowledge of the loads on the arm.

STEP 4: Construct a table of constraints for the particular mechatronic system. The table contains a complete list of the every component of the mechatronic system, the table is to indicate that, when a particular attribute of the component is being modified, which attribute of which component (both within the domain and across the domain) would be affected. In addition, each constraint is labeled as based on function, form, and behavior.

An example of the cross-disciplinary constraints would be the force calculation that is needed for motor selection. The motor that is chosen for the robot arm must not only support weight of the robot arm but also support what the robot will be carrying. To perform the force calculation for each joint, the downward force (weight) of the components that has effect on the moment arm of the joint is multiplied by the linkage length and all the forces are summed to provide the torque required about each joint. This calculation needs to be done for each lifting actuator. In Table 3.5 the cross-disciplinary constraints for the SG5-UT robot arm are identified and listed.

In this robot arm example, constraints based on function are ones that are expressed in equations: torque of the servomotor is related by moment arm and weight of the components that it supports; area and volume of a component is related by the dimensions of the component. Constraints based on form are ones that are related by physical appearance, in this case, geometry: the location of the joint and location of axis of rotation of the component is coaxial. Constraints based on behavior are ones that respond to certain signal: the servomotor controlling the gripper respond to the signal sent by the servo control to determine the motion of the gripper. In Table 3.5, constraints based on function are labeled by “T”, constraints based on form are labeled by “F”, and constraints based on behavior are labeled by “B”.

Table 3.5: Table of constraints for the robot arm

(Note: T = function, F = form, B = behavior)

Component (Attribute)	Constrained attribute within the domain	Constrained attribute across the domain
Bicep (L, W, H) T	Bicep (Weight, Volume)	Shoulder Servo (Torque)
Shoulder Joint (Location) F	Bicep (Location of axis of rotation)	
Forearm (L, W, H) T	Forearm (Weight, Volume)	Shoulder Servo Elbow Servo (Torque)
Elbow Joint (Location) F	Forearm (Location of axis of rotation)	
Wrist (L, W, H) T	Wrist (Weight, Volume)	Shoulder Servo, Elbow Servo, Wrist Servo (Torque)
Base servo (Torque) T		Wheel Plate (Rotation speed)
Servo Power Supply T	All Servos (Torque)	
Servo Controller B	All Servos (Control)	Gripper (Location, velocity)

3.9 Summary

In this chapter the list of constraints are categorized based on function, form, and behavior. Since many of the propagation of design modification between MCAD and ECAD systems are related to constraints based on function, i.e., changing attribute value of one component affect the attribute values of other components through constraint equation, the discussion of constraint modeling and propagation in the next two chapters focuses on constraints based on function.

CHAPTER 4
CONSTRAINT MODELING

This chapter addresses the second phase of this research: constraint modeling. In designing mechatronic systems, design modifications often affect certain attributes of components and therefore require changing components affected by the modified attribute value. For example, the change in dimensions of mechanical components (e.g., size of an elevator, length of a mechanical link) leads to a change in electrical component (e.g., a more powerful electric motor). Rather than going through the process of converting the CAD model into a neutral file (those often are complex and memory consuming because they contain all sorts of information regarding engineering analysis), passing the neutral file to the other domain, and have the engineers in the other domain search through a vast amount of information to find the information about the design modification, a much more direct and efficient approach is to relate the component attributes in the mechatronic system through constraints. One way to model constraints in mechatronic systems is to use an object-oriented modeling language, such as the Unified Modeling Language (UML) or the Systems Modeling Language (SysML).

4.1 UML vs. SysML for Constraint Modeling

4.1.1 Overview of UML

UML has been under development since 1994 by the Object Management Group (OMG). It enables system developers to specify, visualize, and document models in a manner that supports scalability, security, and robust execution (Pender 2003). UML modeling raises the level of abstraction throughout the analysis and design process,

therefore it is easier to identify patterns of behavior and thus define opportunities for refactoring and reuse. UML modeling facilitates the creation of modular designs resulting in components and component libraries that expedite development and help insure consistency across systems and implementations (Pender 2003).

UML is designed specifically to represent object-oriented systems. Object-oriented development techniques describe software as a set of cooperating blocks of information and behavior. For example, a performance at a theater would be coded as a discrete module with its own data about dates and time, and behavior such as schedule, cancel, or reschedule. UML itself is not a method but it is designed to be compatible with the object-oriented software development methods of its time (e.g., Booch, OMT). Most people are familiar with the UML used for software modelling. In this software perspective, the elements of UML map pretty directly to elements in a software system. Another way of looking at UML is with the conceptual perspective, in which UML represents a description of the concepts of a domain of study (Fowler 2004).

The recent version of UML, UML 2.0, has 13 types of diagrams (see Figure 4.1)

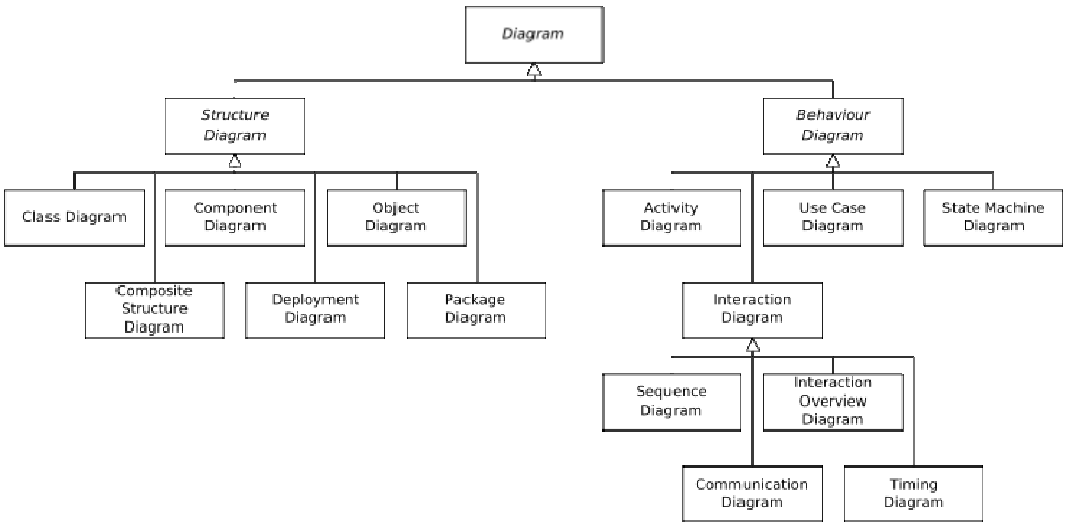


Figure 4.1: UML diagram types (Source: Wikipedia)

UML diagrams represent three different views of a system model:

- Functional requirements view: emphasizes the functional requirements of the system from the user's point of view, such as use case diagrams.
- Static structural view: emphasizes the static structure of the system using objects, attributes, operations, and relationships, such as class diagrams.
- Dynamic behavior view: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects, such as sequence diagrams, activity diagrams, and state-machine diagrams.

Class diagrams can be considered the most commonly used diagram in UML. It is widely used because it can be applied to a wide range of modelling concepts. A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them. In a class there are properties and operations. Properties represent structural features of a class. Properties are a single concept, but they appear in two distinct notations: attributes and associations. Attributes describe a property as a line of text within the class box. For example, in a class box titled "motor", the attributes in that class box can be weight and maximum torque.

The other way to notate a property is through association. An association is a solid line between two classes, directed from the source class to the target class. For example, if both "motor" and "power supply" are classes, then an association called "provide_electric_energy" is drawn between them. In general, attributes are used for representing properties within a class, such as motor having properties of weight and torque, and associations are used for relationship between classes, such as between motor

and power supply. Also, associations are either unidirectional or bidirectional, and carries multiplicity as an indication of how many objects may fill the property. Operations are the actions that a class carries out. The full UML syntax for operations has the following: visibility which marks public or private, name, list of parameters, the type of the returned value, and property values.

An *object diagram* is a snapshot of the objects in a system at a point in time. It is used to show an example configuration of objects, which can be very useful when the possible connections between objects are complicated. In many situations, one can define a structure precisely with a class diagram, but the structure is still difficult to understand. In these situations, providing some object diagrams can provide a much better illustration.

Activity diagrams are used to describe procedural logic, business process, and work flow. They are similar to flowcharts; the principal difference between them and flowcharts is that they support parallel behavior. The activity diagram allows whoever doing the process to choose the order in which to do things, the diagram merely states the essential sequencing rules that need to be followed. This is useful for concurrent algorithms in which independent threads can do things in parallel.

A *sequence diagram* captures the behavior of a single scenario. The diagram shows a number of example objects and the messages that are passed between these objects within the use case. A package diagram allows the grouping of elements in any UML constructs into higher-level units. It is most commonly used to group classes. Package diagrams are useful on large-scale systems for obtaining a picture of the dependencies between major elements of the system. Use cases are used for capturing the functional requirements of a system. Use cases work by describing the typical

interactions between the users of a system and the system itself and thereby providing a narrative of how a system is used.

The table below illustrates how UML representations can be used to describe a mechatronic system and its components. The examples shown in the table are description of servos.

Table 4.1: UML representations for use in mechatronic systems

UML representation	Description	Example
Class	A family of components of same nature.	Servos
Object	A particular component.	HS-645MG Super Torque Gear Servo
Attributes	A property of the component.	Torque, Weight, Dimension
Attribute's value	A value or data to that particular property of the component.	0.15m by 0.07m by 0.10m 14 N , 0.5 N*m
State	A condition of the component.	Running at 6V
Operation	An event or occurrence of the component.	Rotating the elbow link

4.1.2 An Alternative Approach for Modeling Constraints: SysML

Although UML can provide a solid model of the properties of a mechatronic system, it is not an ideal candidate for constraint modeling due to one of its fundamental

gap: the ability to perform engineering analysis. As a result of using modeling languages like UML, non-standardized engineering analysis models are often disjoint from the system architectural models that specify behavioral and structural aspects of a system. The lack of integration and synchronization between the system architectural models and the engineering analysis models is aggravated as the complexity, diversity, and number of engineering analysis models increases (Peak et al. 2007).

SysML, on the other hand, has the capability to model engineering analysis. It is a general-purpose graphical modeling language with computer-sensible semantics. Parametrics are not part of UML and were introduced as a new modeling capability as part of SysML (Peak et al. 2007). Parametrics address the gap that UML possesses of modeling engineering analysis and provide the mechanism to integrate engineering analysis models with system requirements and design models for behavior and structure. In addition, parametric is a constraint representation that can be used more generally to capture other types of knowledge beyond support for engineering analysis.

The next section provides a more detailed overview of SysML.

4.2 The System Modeling Language (SysML)

The official Object Management Group (OMG) website describes SysML as follows (OMG 2007):

> "The OMG Systems Modeling Language (OMG SysML™) is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. In particular, the language provides graphical representations with a semantic foundation for modeling system requirements,

behavior, structure, and integration with a broad range of engineering analysis. SysML represents a subset of UML2 with extensions needed to satisfy the requirements of the UML™ for Systems Engineering RFP. SysML uses the OMG XML Metadata Interchange (XMI®) to exchange modeling data between tools, and is also intended to be compatible with the evolving ISO 10303-233 systems engineering data interchange standard."

SysML provides model driven approaches to system engineering, moving from Document centric to Model centric. By using SysML one can create an integrated system model that addresses multiple aspects of a system. In Figure 4.2 below a system model of a car is shown.

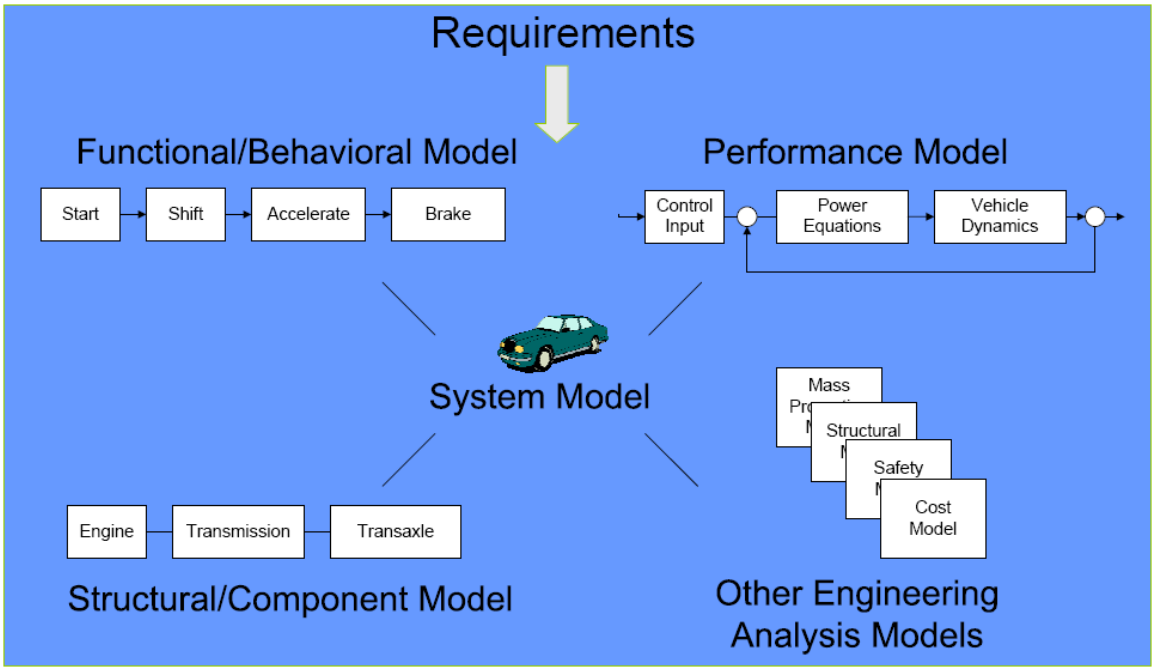


Figure 4.2: System model of a car (OMG 2006)

Using SysML one can model functions and behaviors of the system, in the car example, the functions are start engine, shift gear, accelerate, brake, turn off engine, etc. All these functions can be modeled in Activity diagrams and State-machine diagrams in

SysML. The Block-definition diagram allows modeling of the structure and component of the model. Beside modeling the structure and component of a system, and modeling the function and behavior of a system, SysML also has Parametric Diagrams that can be used to express constraints and equations between value properties. Parametric diagram represents the usage of the constraints in an analysis context, binding the constraint usage to value properties of blocks, for example, vehicle mass is bound to F = m*a.

In summary, by enabling an electronic representation of the product design and its relevant engineering analytical information, SysML "*opens the door to analytics for faster and more effective decision-making across the entire systems development lifecycle.*" (Balmelli 2006).

4.2.1 Modeling Structure with Blocks

A *block* is the modular unit of structure in SysML that is used to define a type of system, system component, or item that flows through the system. A block describes a set of uniquely identifiable instances that share the block's definition (Friedenthal et al. 2008). The block definition diagram is used to define block characteristics in terms of their structural and behavioral features, and the relationships between the blocks.

A block can define the following (Friedenthal et al. 2008): a type of a logical or conceptual entity, a physical entity (e.g., a system); a hardware, software, or data component; a person; a facility; an entity that flows through the system (e.g., water); or an entity in the natural environment (e.g., the atmosphere or ocean). Blocks are often used to describe reusable components that can be used in many different systems. In Figure 4.3 the block definition diagram of a vehicle is shown.

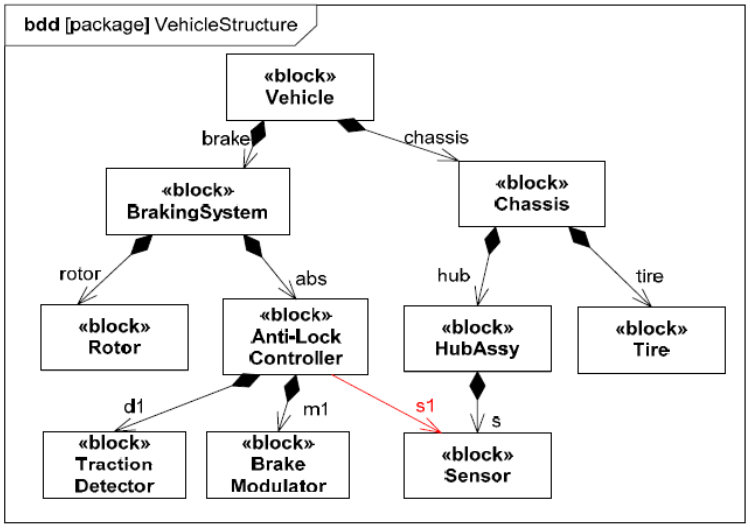


Figure 4.3: Block definition diagram showing structure of a vehicle (OMG 2006)

Properties are one category of features of a block. They are used to capture the structural relationships and values of a block. A property has a type that may be another block, or some more basic concept such as an integer value. There are three categories of properties:

- Part properties: describe the decomposition of a block into its constituent elements.
- Reference properties: describe weaker relationships between blocks than the composition relationship represented by part properties.
- Value properties: describe the quantifiable characteristics of a block, such as its weight and velocity.

4.2.2 Modeling Constraints Using Parametric Diagrams

The greatest advantage of using SysML in this thesis is the ability of SysML to support modeling constraints on the performance and physical properties of systems and their environment to support a wide array of engineering analyses. Parametric models in SysML can capture the constraints on the properties of the components in the

mechatronic system, which can then be evaluated by relevant analysis tool. The constraints are expressed as equations whose parameters are bound to the properties of a system. Each parametric model can capture a particular engineering analysis of the design.

SysML introduces a constraint block to support the construction of parametric models. A constraint block is a special kind of block used to define equations so that they can be reused and interconnected. Constraint blocks have two main features: a set of parameters and an expression that constrains the parameters. The definition and use of constraint blocks is represented on a block definition diagram and parametric diagram, respectively.

Block definition diagrams are used to define constraint blocks in a similar way to which they are used to define blocks, as described in the previous section. Figure 4.4 shows the block definition diagram that is used to define the parameters in vehicle dynamics analysis. Parametric diagrams are used to create systems of equations that can constrain the properties of blocks. The diagram frame of a parametric diagram represents either a block or a constraint block. Figure 4.5 shows the parametric diagram that is used to construct systems of equations for vehicle dynamics analysis.

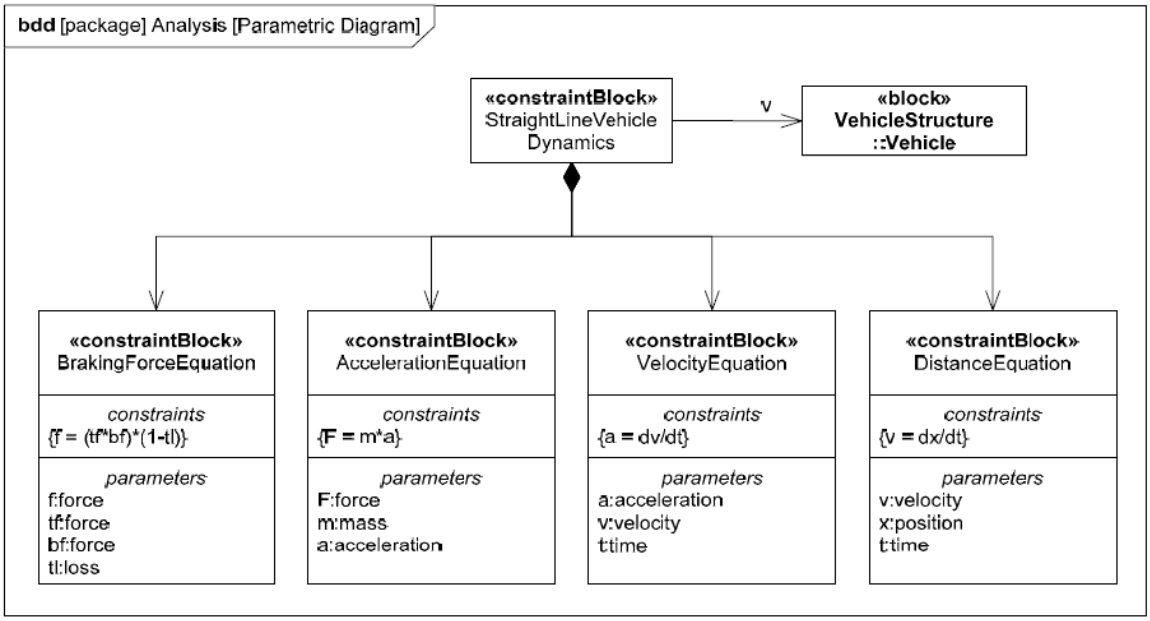


Figure 4.4: Block definition diagram for parameters used in engineering analysis (OMG2006)

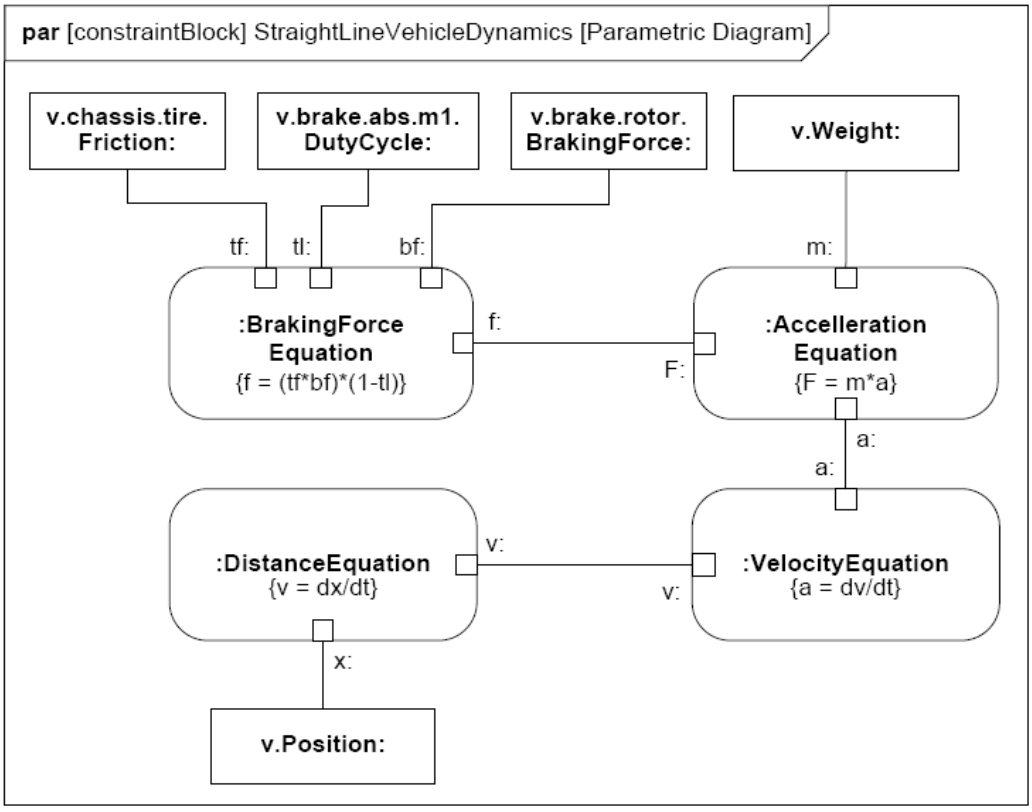


Figure 4.5: Parametric diagram for vehicle dynamics analysis (OMG 2006)

SysML includes a generic mechanism for expressing constraints on a system as text expressions that can be applied to any model element. SysML does not provide a built-in constraint language because it was expected that different constraint languages, such as OCL, Java, or MathML, would be used as appropriate to the domain (Friedenthal et al. 2008). SysML also features a constraint block that extends the generic constraint concept. A constraint block encapsulates a constraint to enable it to be defined once and then used in different contexts. The constraint expression can be any mathematical expression and may have an explicit dependency on time, such as a time derivative in a differential equation. In addition to the constraint expression, a constraint block defines a set of constraint parameters, which is a special kind of property used in the constraint expression. Constraint parameters are bound to other parameters and properties of the blocks where they are used.

In the constraint block, each parameter has a type that defines the set of values that the parameter can take. Typically parameters are scalars, vectors, or a structured data type such as complex. Through its type, the parameter can also be constrained to have a specific unit and dimension. Parameters can also support probability distributions.

The block definition diagram in the constraint model shows all the parameters that are used in the constraint relationships, however, it does not show all the required information needed to interconnect its constraint properties. Specifically, its does not show the relationship between the parameters of constraint properties and the parameters of other constraint blocks. This additional information is provided in the parametric diagram using binding connectors. Binding connectors express equality relationships between their two ends.

In summary, in order to model constraint between the parameters of a system, constraint blocks are used. Constraint blocks model constraints on the properties of blocks to support engineering analyses, such as performance, reliability, and mass properties analysis. The parametric diagram shows how constraint properties bind to one another and to the value properties of blocks through their parameters. They are bound using connectors that express equality between values of the parameters or properties at their ends.

4.3 A Constraint Model for Mechatronic Systems

The Parametric Diagram in SysML with its capability to perform engineering analysis makes it a very suitable candidate for constraint modeling and propagation for mechatronic systems. In Figure 4.6 below is a scheme of constraint model that will be used later for constraint propagation.

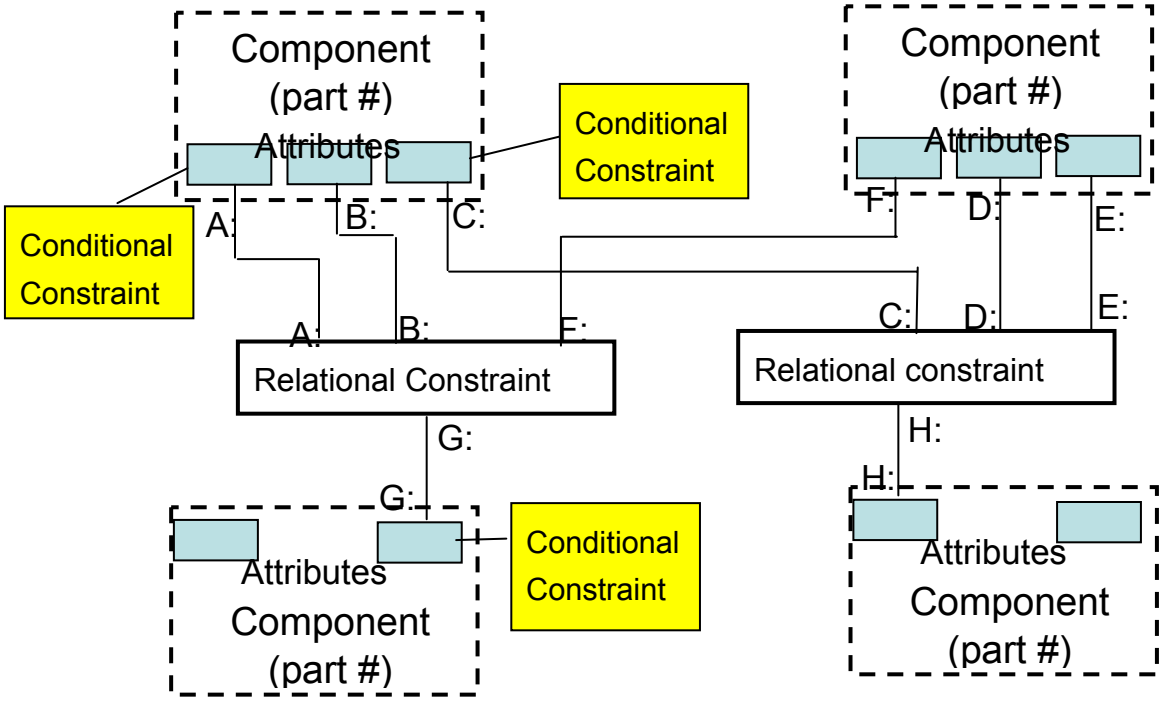


Figure 4.6: Constraint model for mechatronic systems

There are three main components in this constraint model: component block, relational constraint, and conditional constraint.

Component Blocks

A Component block specifies a component inside the mechatronic system, its part number identifying the exact component, and the specific attributes that particular component has. In Figure 4.7 below the usage of a component block is shown.

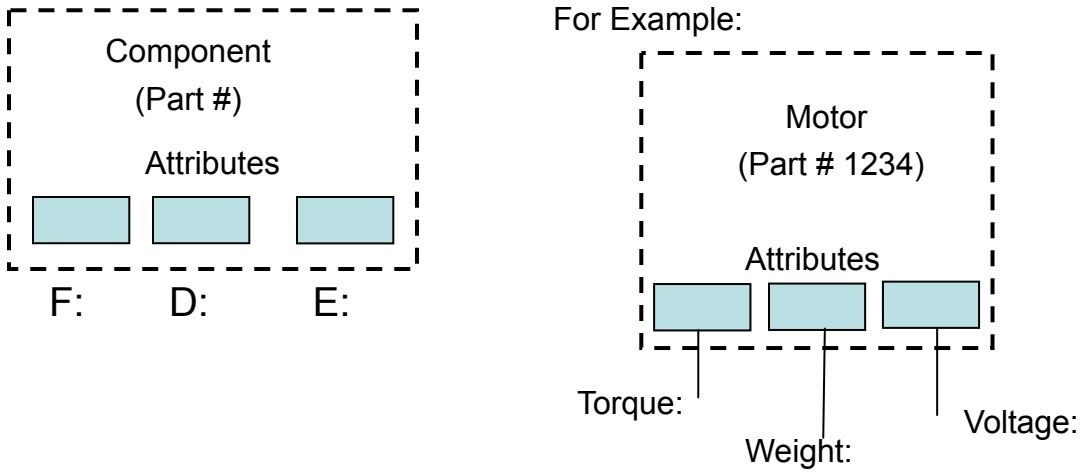


Figure 4.7: Component block and the motor example

Conditional Constraint

A conditional constraint specifies the limit values of a particular attribute, due to design specification, physical limits of the components, etc. In Figure 4.8 the usage of a conditional constraint in the constraint model is shown.

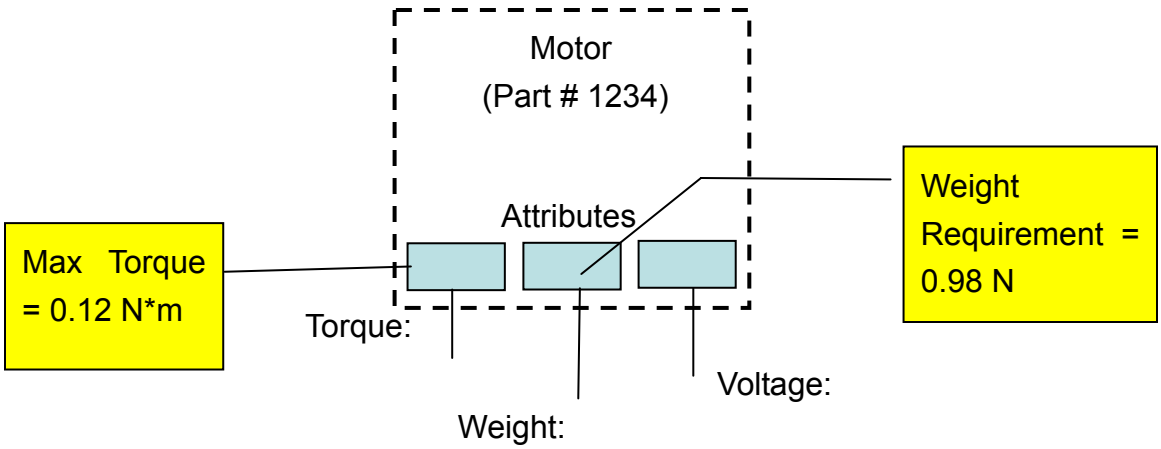


Figure 4.8: Conditional constraint in the motor example

Relational Constraint

A relational constraint is specified in an equation block. It is the constraint that relates the attributes of the components in the mechatronic system. In Figure 4.9 the usage of a relational constraint in the constraint model is shown. In Figure 4.9 A, B, F, G all are attribute values of some components in the mechatronic system, and the relationship between these attribute values is defined by the equation in the equation block.

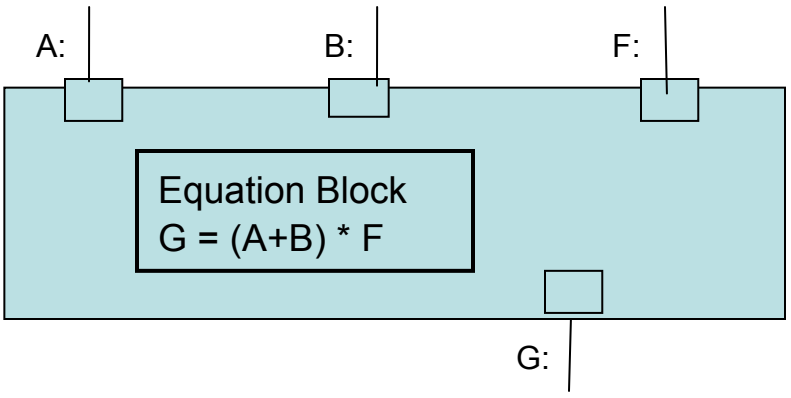
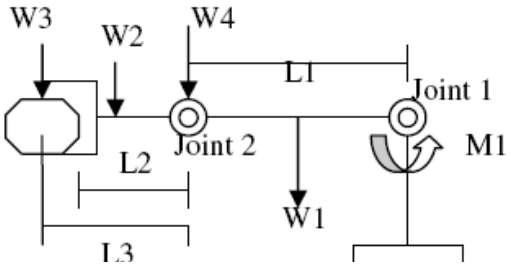


Figure 4.9: Example of a relational constraint

4.4 Illustrative Example: The Constraint Model for a Robot Arm

Figure 4.10 below displays the torque calculation for a simple robot arm with two degrees of freedom. In modeling this mechatronic system, the torque (the torque that the motor is required to provide for the robot arm) is an electrical attribute specified in the ECAD model, while arm length and weight are mechanical attributes specified in the MCAD model.



Torque about joint 1:

$$M1 = \frac{L1}{2} * W1 + L1 * W4 + (L1 + \frac{L2}{2}) * W2 + (L1 + L3) * W3$$

Torque about joint 2:

$$M2 = \frac{L2}{2} * W2 + L3 * W3$$

Figure 4.10: Torque calculation about the joints of a robot arm

In Figure 4.11 the constraint blocks defining the equations and parameters for the motor torque calculation is shown. In Figure 4.12 the constraint model with the constraint blocks, conditional constraints, and relational constraints defined for this particular robot arm is shown.

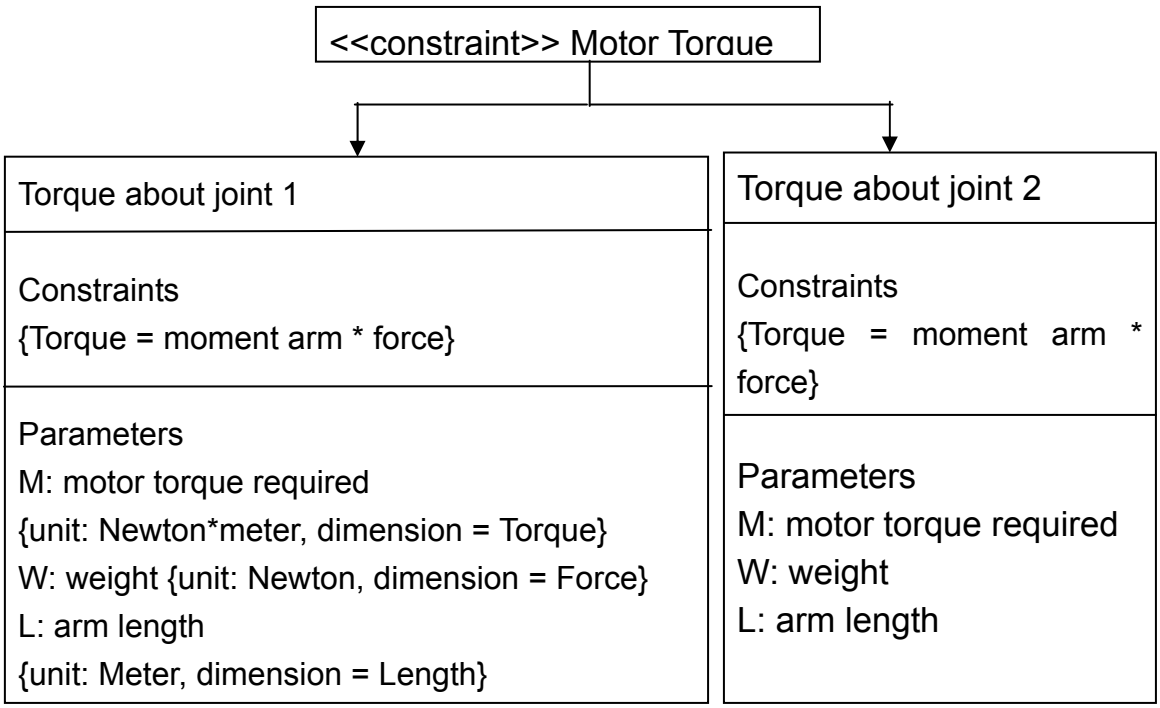


Figure 4.11: Constraint blocks with parameters.

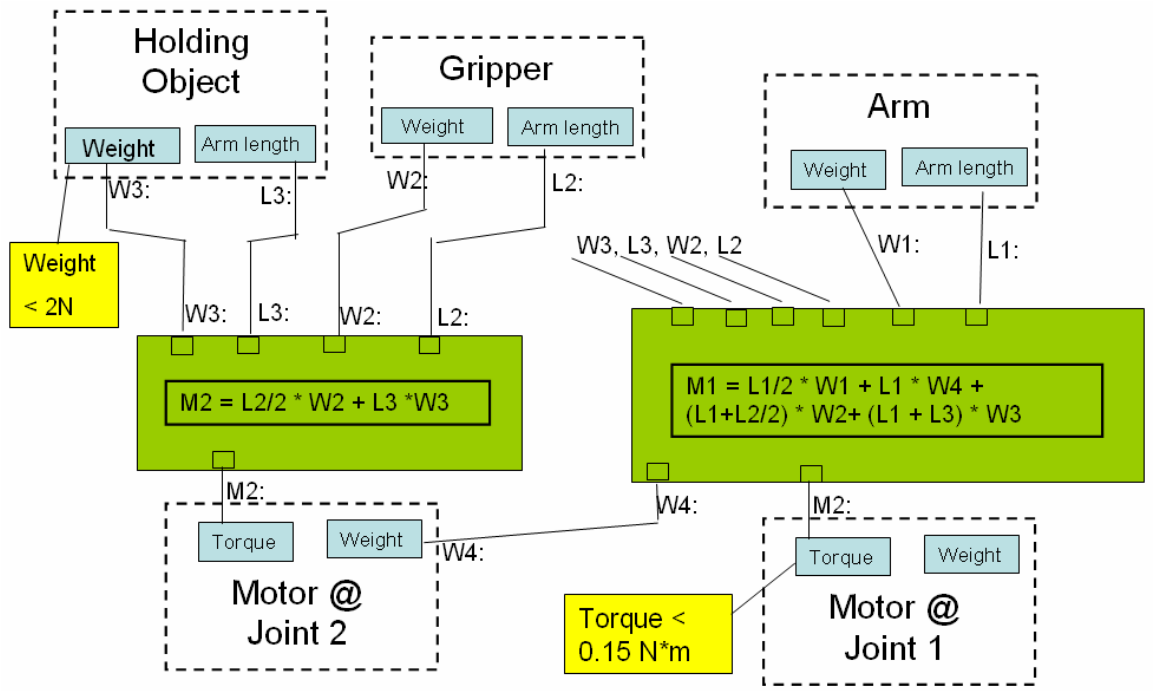


Figure 4.12: Constraint model with parametrics for the robot arm

From the constraint model, one can see that by changing the attributes of the holding object (such as requiring to holding something heavier), or the attributes of the gripper, any of these change would affect the motor at joint 2. And since the weight of the motor at joint 2 is also part of the relational constraint to the motor at joint 1, the attribute of the motor at joint 1 would also be affected.

4.5 Summary

In this chapter two object oriented modeling languages are explored: UML and SysML. While UML have the capability to model various aspects of a mechatronic system, it lacks the capability to perform engineering analysis. SysML, on the other hand, has the capability to perform engineering analysis through defining parameterics and constraints in the Parametric Diagram, and hence SysML is an ideal candidate for use in the constraint modeling of mechatronic systems. The value of using SysML is not just limited to modeling constraints based on functions, its various other diagrams (Activity Diagram, State-machine Diagram, etc.) may also be very useful for modeling the other two classes of constraints: constraints based on forms and based on behavior.

A modeling scheme based on SysML is developed for modeling constraints in mechatronic systems. In this scheme there are three parts: component blocks that allow defining of components and their attributes, conditional constraint for setting limitations on the attribute values, and relational constraint for defining relationship between attributes of components through an equation block. An illustrative example of using this constraint model to model torque calculation about the joints of a robot arm is provided. The next chapter illustrates the use of this constraint model in the implementation of constraint propagation between EPLAN Electric P8 and SolidWorks.

CHAPTER 5
CONSTRAINT PROPAGATION

This chapter addresses the third phase of this research: constraint propagation. In the previous chapter, a constraint model is presented. In the constraint model, the designer can create a component using the component block and create attributes to assign specific attribute values. The designers can create conditional constraints to set the limit for the attribute value and relational constraints to link the attribute value of one component to the attribute values of other components. In this chapter, the constraint model is applied to the implementation to achieve propagation of design modification.

5.1 Constraint Propagation Flow Diagram

The flow diagram (see Figure 5.1) illustrates the process of constraint propagation. When a design modification is made in either domain, first check the conditional constraint to ensure that such modification is allowed within the design specification. Then perform calculation to calculate the new attribute values for all components based on the relational constraints specified in the constraint model. If all the new attribute values satisfy all the pre-defined conditional constraint, then send a change request to the other domain (i.e. mechanical to electrical or vice versa). The change request should specify the change that has been made and the impacts on the model. Once the engineer on the other side accepts the change, the constraint propagation is successfully completed. If at any point the conditional constraint is violated, or that at the end the engineer from the other side rejects the change request, the engineer who initiated the design modification is notified and the change is not allowed.

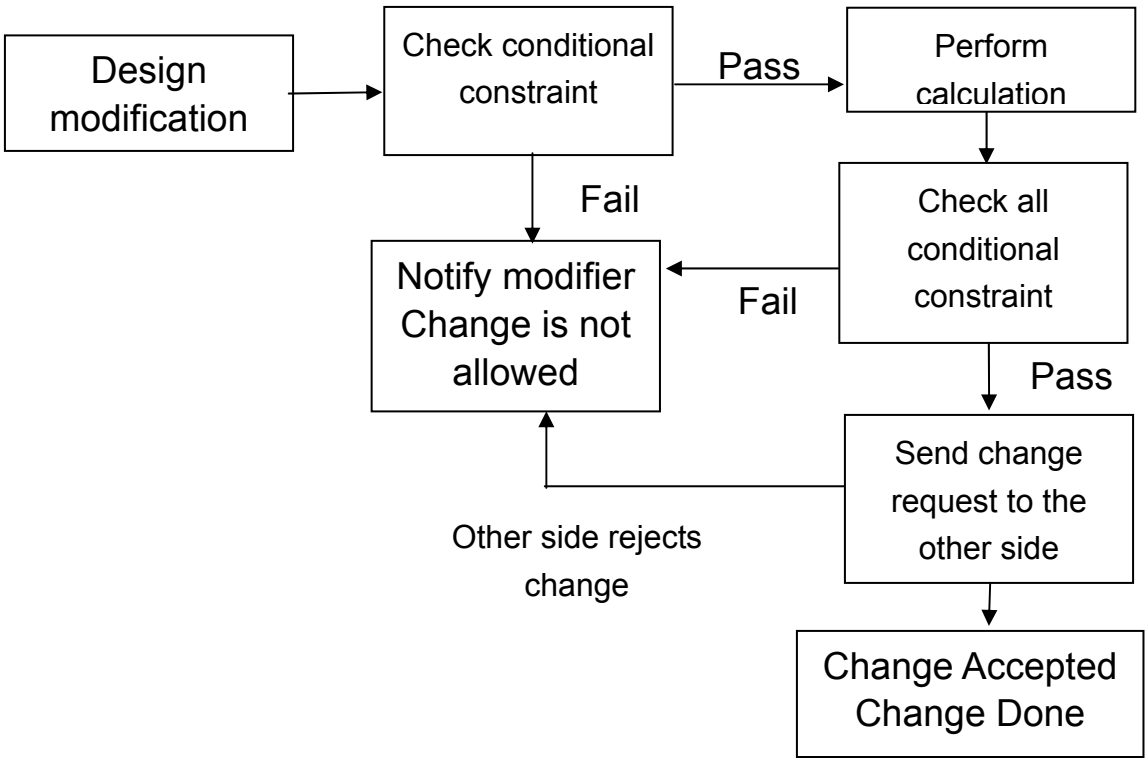


Figure 5.1: Constraint propagation flow diagram

5.2 Implementation

Functions are created using both EPLAN API (VB .NET) and SolidWorks API (a built-in VB editor) to send and retrieve data through Excel. The implementation is as follows: define the constraint equations and parameters in Excel. Create two buttons: "submit change" and "retrieve data" in both CAD systems. Click on "submit change" the new data will be uploaded to a given excel file. When the other side click "retrieve data", new data is shown with a message asking the user to accept or reject changes. Either case, in the excel file there is a location storing accept/reject response to keep both side informed. So whether the receiver accept or reject the change, the sender will know when "retrieve data" function is selected. Shown in Figure 5.2 is an overview of the implementation.

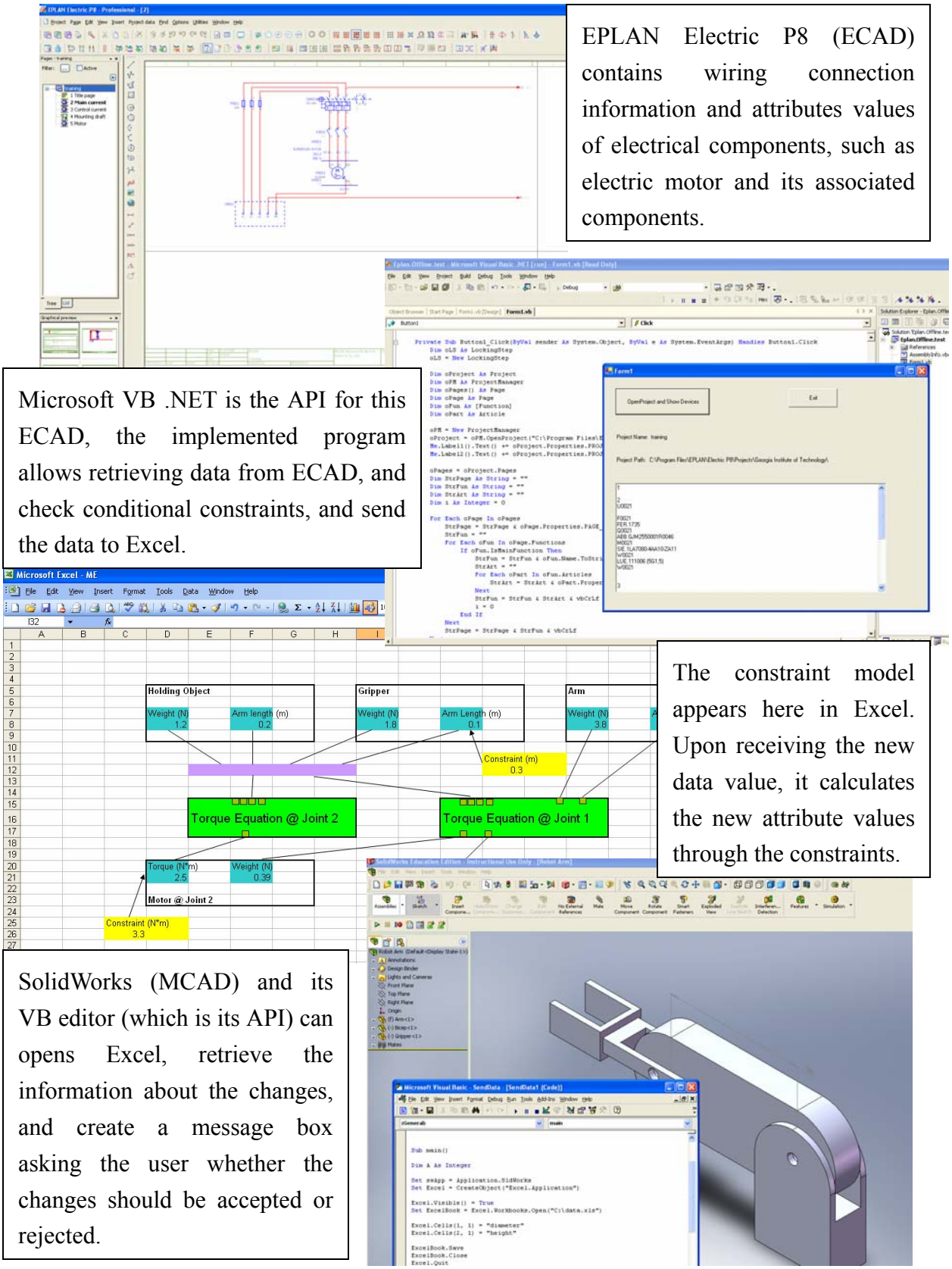


Figure 5.2: Implementation Overview

As discussed in the last chapter, the constraint model has component blocks displaying the components in the mechatronic system, conditional constraint specifying the limitation of attribute values, and relational constraint in the form of equations specifying the relationships between attributes of the components.

Figure 5.3 below illustrates a constraint model for a robot arm. This constraint model is created for torque calculation for the motor selection at the joint of the robot arm. In this constraint model there are five component blocks, some represent components from MCAD model and others represent components from ECAD model. The mechanical component blocks have attributes of weight and arm length, both of which are needed for torque calculation. On the electrical side the attributes are torque and weight (of the motor).

The conditional constraints have two purposes. One is to prevent the designer from accidently entering a number for a component attributes that turns out to be nonsensical. And the other one is that they may exist as a result of design specification, for example, the specified arm length of 0.6m and 0.3m is to prevent collision with other objects during operation, and the torque of the motor cannot be greater than certain value to prevent the robot arm from moving too fast.

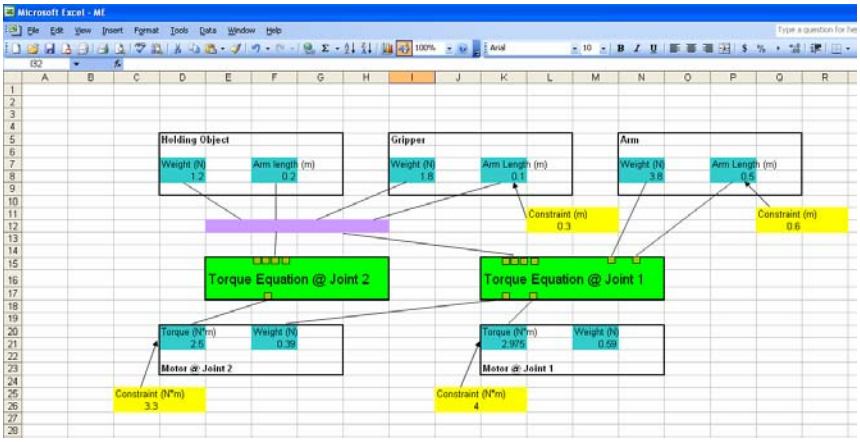


Figure 5.3: Constraint Model of Robot Arm in Excel

5.3 Four Case Scenarios

Case Scenario 1: Failed Initial Conditional Constraint Check

The designer makes a modification, but the modification does not satisfy the conditional constraint, as shown in the constraint propagation flow diagram below:

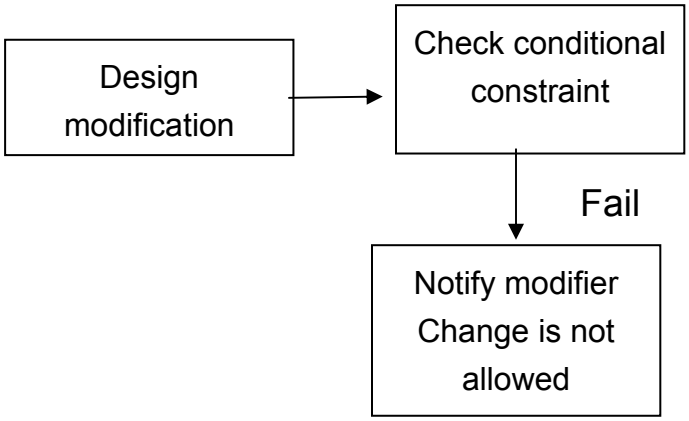


Figure 5.4: Flow diagram for case scenario 1

In this example, the robot arm's arm length was 0.25m, the designer modified it to 0.5m, however, the conditional constraint is set at 0.4m, and hence error message appears.

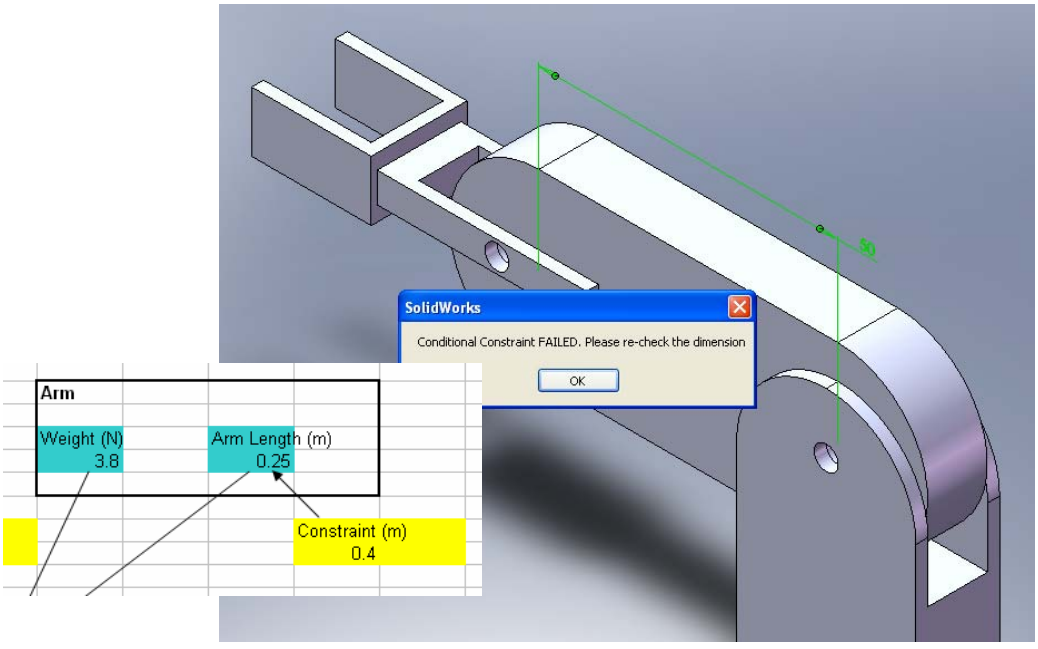


Figure 5.5: Case scenario 1 screen shots (Mechanical side)

Similar situation can also occur on ECAD side. The designer decided to change the electric motor from motor with torque of 0.4 N*m (Part number “LENZE.080.11”) to a motor with torque of 0.8 N*m (Part number “LENZE.080.32”), however the conditional constraint for the motor is set at 0.7 N*m, and hence error message appears.

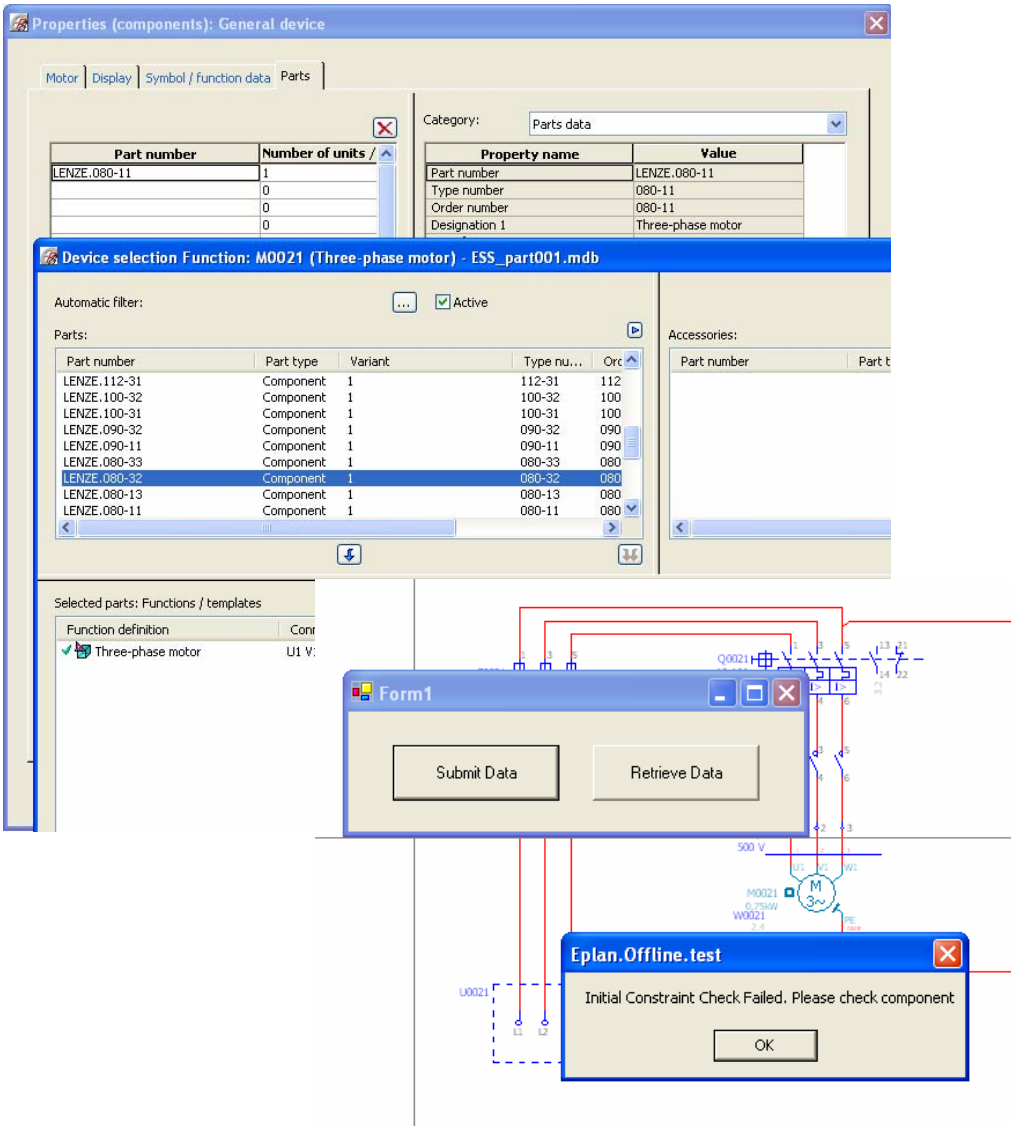


Figure 5.6: Case scenario 1 screen shots (Electrical side)

Case Scenario 2: Failed Conditional Constraint Check after Calculation

The designer makes a modification and the modification satisfies the conditional constraint, however, the other attribute value affected by this modification does not satisfy conditional constraint, as shown in the constraint propagation flow diagram below:

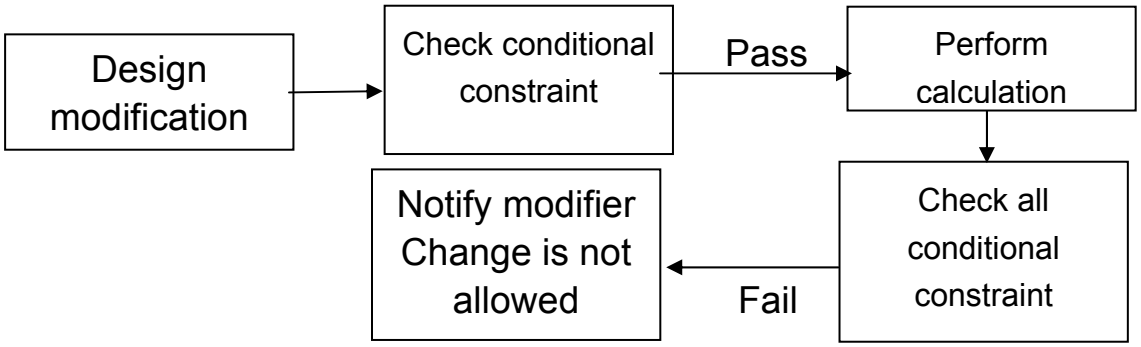


Figure 5.7: Flow diagram for case scenario 2

In this example, the robot arm's arm length was 0.35m, among with other components would require a torque of 2.1815 N*m for the motor at joint 1. The designer modified it to 0.5m, which satisfy the conditional constraint set at 0.6m, however, this creates a torque requirement of 2.975 N*m, and conditional constraint for the torque is set at 2.5 N*m, and hence error message of "Initial Check Passed. Constraint Failed during Propagation" appears. In Figure 5.8 the screen shots of such scenario case is displayed.

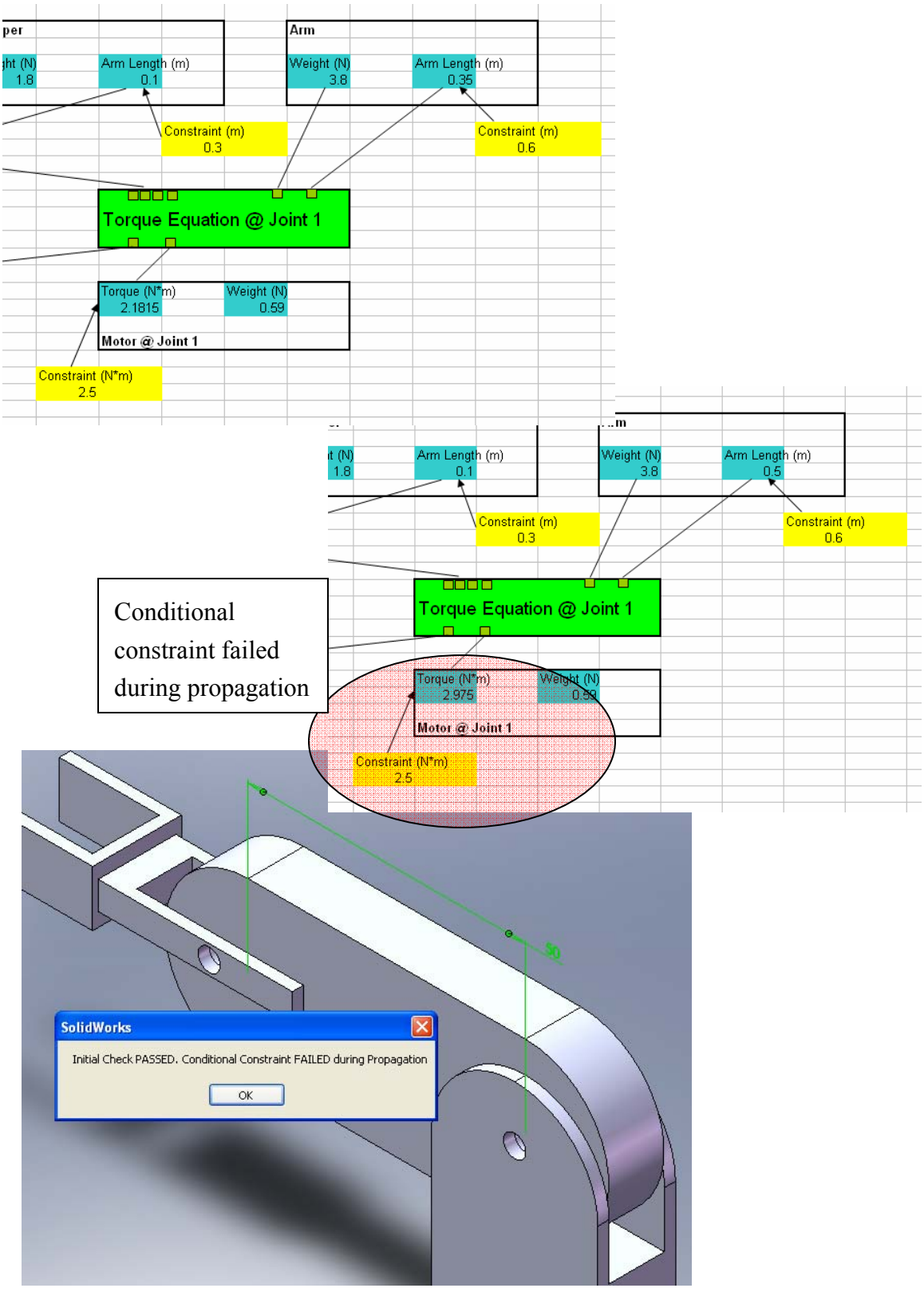


Figure 5.8: Case scenario 2 screen shots

Case Scenario 3: Other Side Rejects Change

The designer makes a modification and the modification satisfies all conditional constraints, however, the other side decided for whatever reasons to reject the request of change, as shown in the constraint propagation flow diagram below:

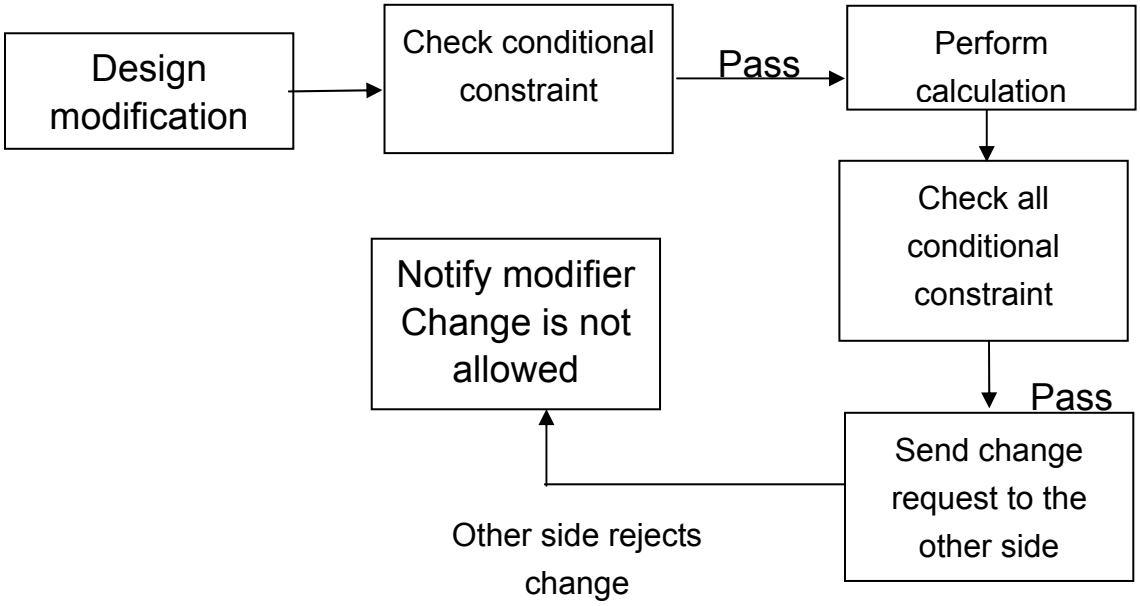


Figure 5.9: Flow diagram for case scenario 3

In this example, a design modification was made on the MCAD side. The design modification passed all the conditional constraints. When “Retrieve Data” is performed on the ECAD side, a request message would appear on the ECAD side requesting to change the original motor “LENZE.080-11” to a new motor “LENZE.080-13”. The engineer on the ECAD side decided to reject this change request. An input-box for reason of rejection appears, and the engineer entered “Too much power consumption”. When the MCAD engineer retrieved data, the rejection message and the reason would appear. In Figure 5.10 the screen shots of such scenario case is displayed.

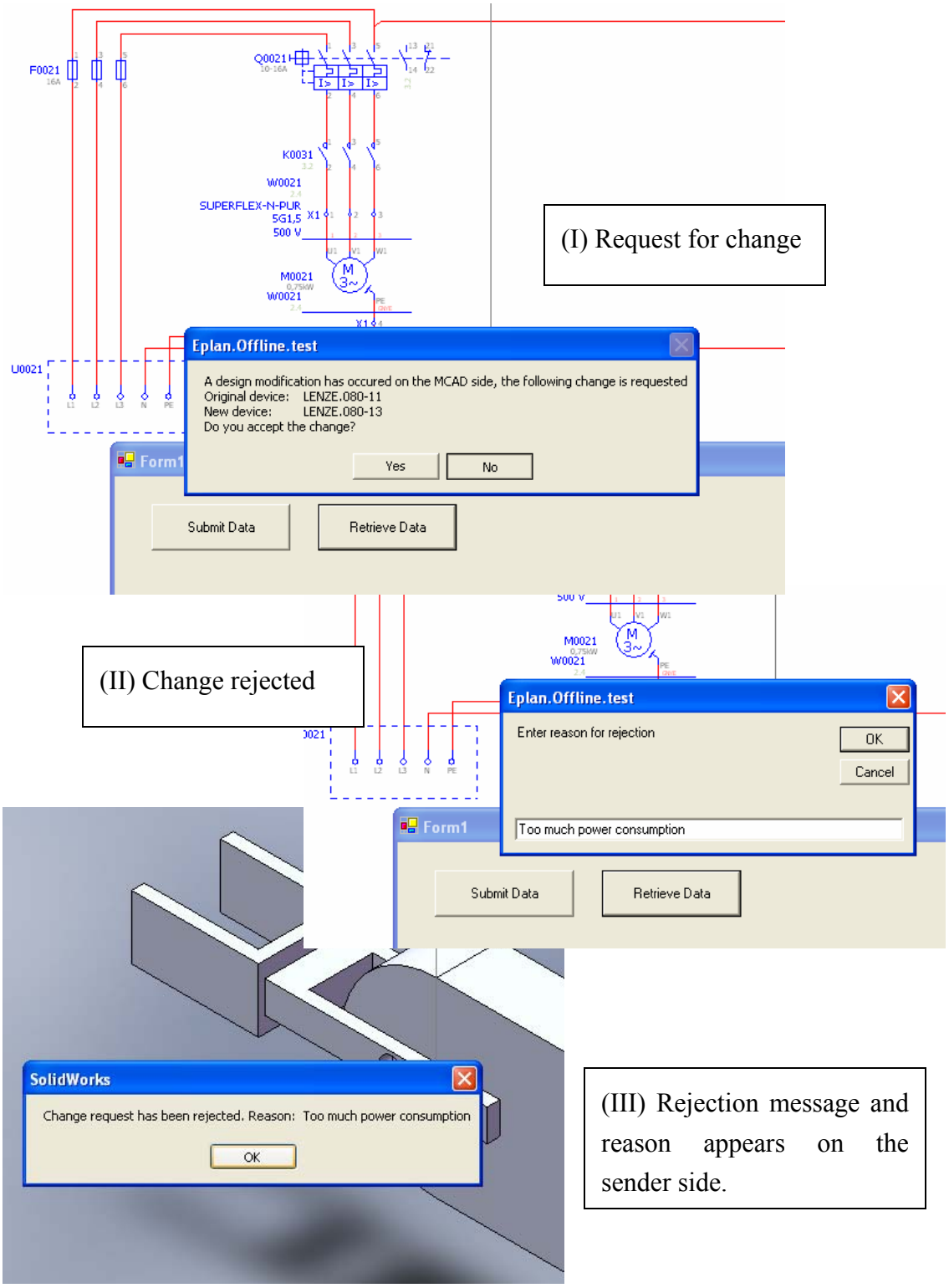


Figure 5.10: Case scenario 3 screen shots

Case Scenario 4: Other Side Accepts Change

The designer makes a modification and the modification satisfies all conditional constraints, the other side decided to accept the change and thus the design modification has been successfully propagated, as shown in the constraint propagation flow diagram below:

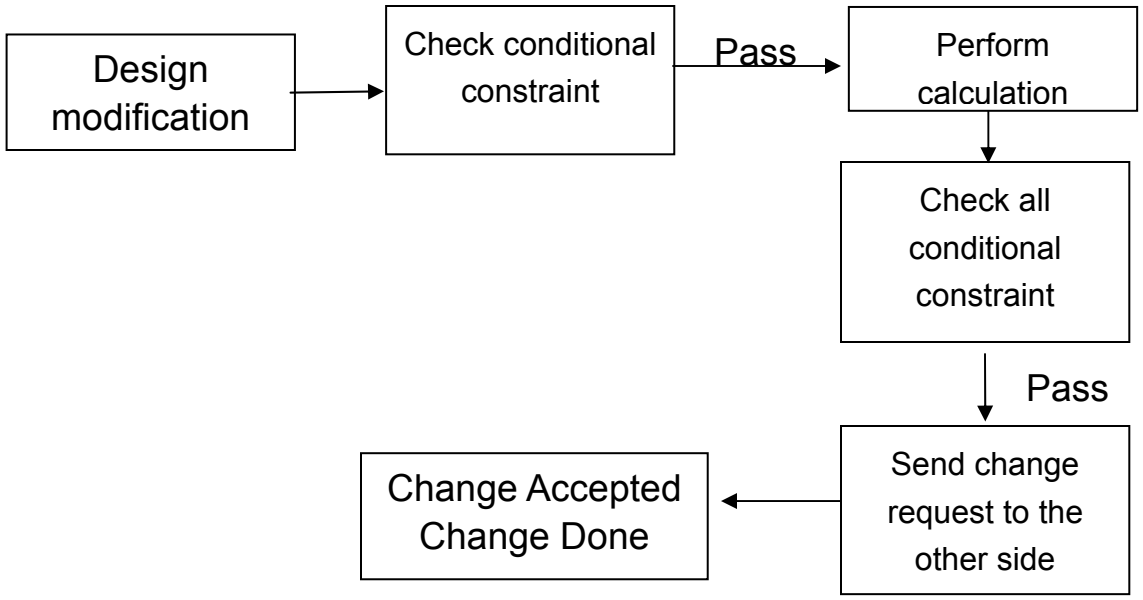


Figure 5.11: Flow diagram for case scenario 4

In this example, a design modification was made on the ECAD side. The design modification passed all the conditional constraints. When "Retrieve Data" is performed on the MCAD side, a request message would appear on the MCAD side requesting to change the arm length of the robot from 0.5m to 0.7m. The engineer on the MCAD side decided to accept the change. The dimension change would occur on the MCAD model. When the engineer on the ECAD side performed "Retrieve Data", a message saying "Design Modification Accepted" would appear. A successful propagation of design modification is completed. Figure 5.12 shows the screen shots of such scenario case.

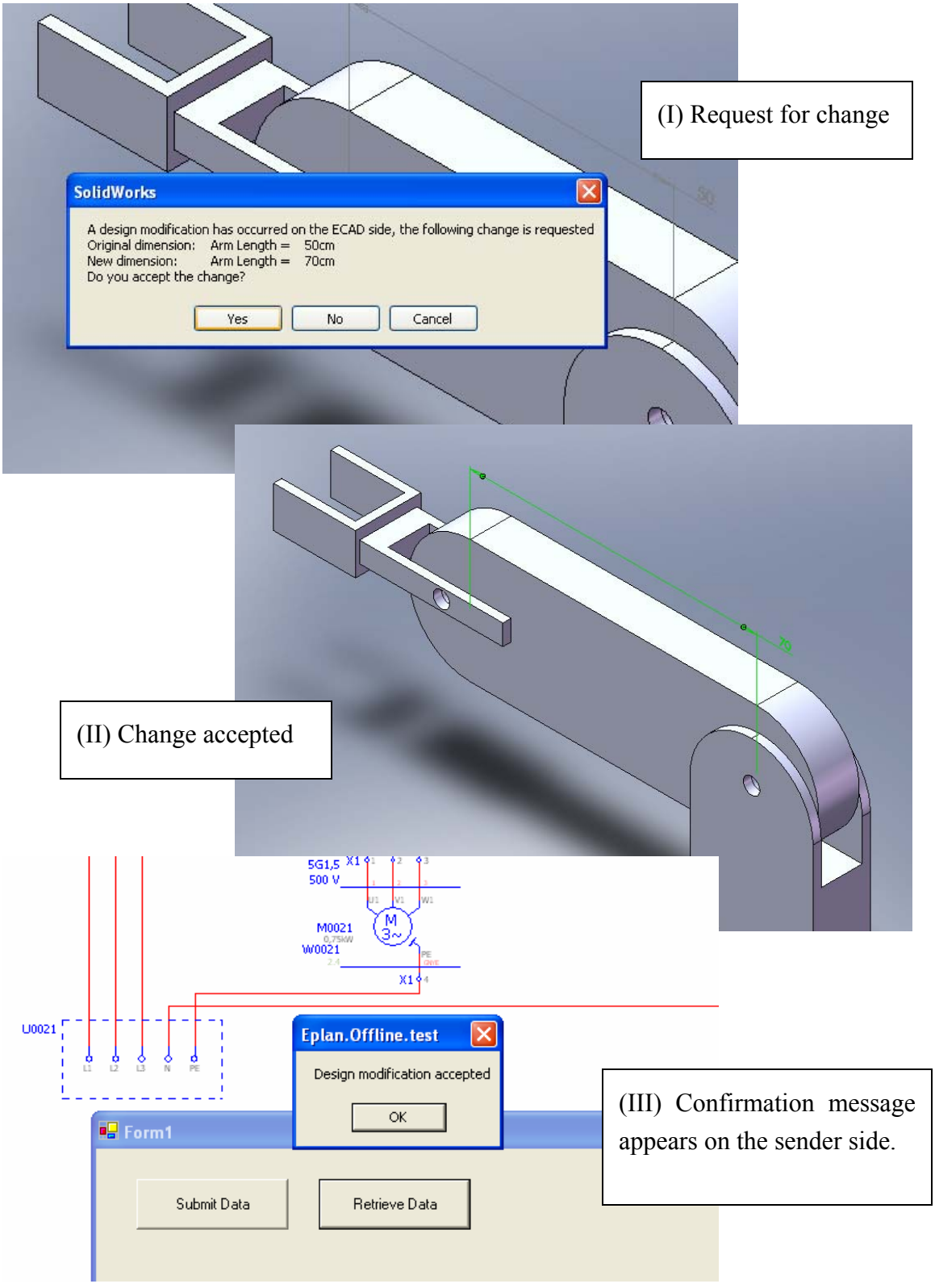


Figure 5.12: Case scenario 4 screen shots

Sometimes it is required to replace more than just one component on an ECAD model. For example, when the motor gets replaced, the motor overload switch also needs to be replaced to suit the new motor. In EPLAN Electric P8, the user has the opportunity to create a window to enclose multiple electrical components, and save that configuration as a variant. For example, motor "LENZE.080-11" and its motor overload switch is saved as "Variant A", motor "LENZO.080-13" and its motor overload switch as "Variant B", and so on. Through its API, EPLAN provides access to a class called "WindowMacro" that enables users to create, load and save window macros, define or obtain the size and position of so-called macro boxes that may contain part variants, and select existing part variants to be placed within these boxes. As for the actual implementation of this in the context of this thesis, users may load a window macro by using the "Open" method of WindowMacro class, which takes two parameters: project name and macro file name. Then the "ChangeCurrentVariant" method of this class provides the means to change a current macro representation type and a specific variant assigned to this macro.

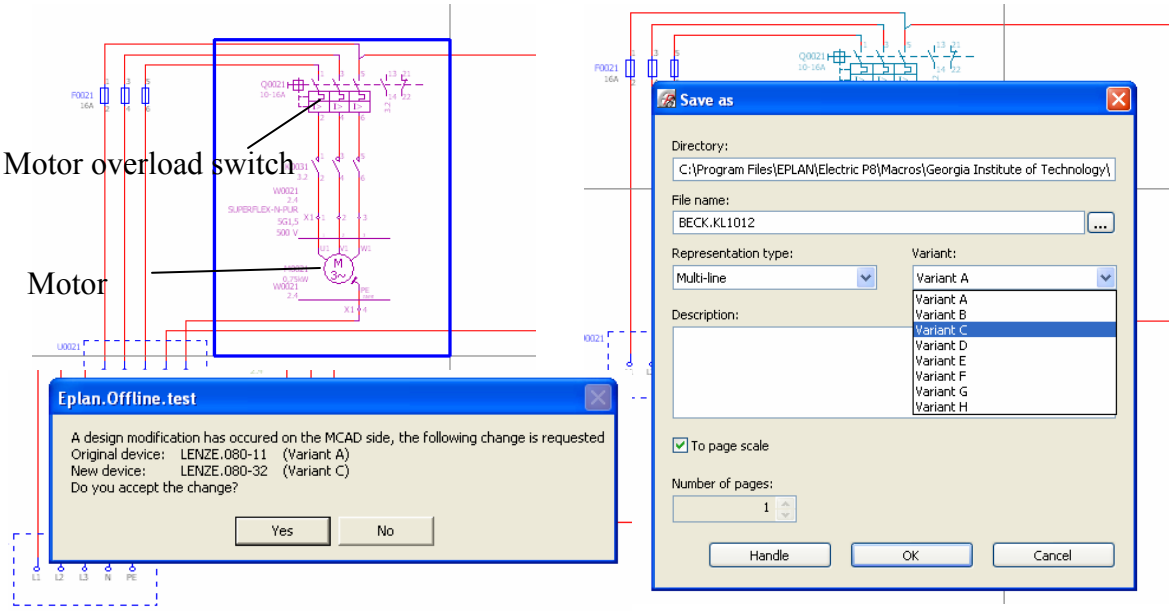


Figure 5.13: Case scenario 4 with variant selection screen shots

5.4 Summary

The constraint propagation flow diagram specifies four scenarios. The first scenario is the design modification fails the initial conditional constraint check and thus propagation is not allowed. The second scenario is the design modification passes the initial conditional constraint check but the new resulting attribute values due to this modification fail their conditional constraint and thus propagation is not allowed. The third scenario is the design modification passes all conditional constraint check and a request for modification appear on the other side, however, the designer on the other side decides to reject the change and thus the design modification is not allowed and the sender receives the reason of change rejection. The fourth scenario is the design modification passes all conditional constraint and a request for modification appear on the other side, the design on the other side accepts the change and thus the design modification is successfully carried out and the sender receives a confirmation.

The implementation involves communication between EPLAN Electric P8 (ECAD software), Application Programming Interface (API) of EPLAN Electric P8, SolidWorks, API of SolidWorks, and Excel. Visual Basic language is used in the implementation. Functionalities are created in the API such information are sent to and retrieved from the CAD software and the Excel spreadsheet.

CHAPTER 6
CONTRIBUTIONS AND CLOSURE

6.1 Evaluation of Research Questions and Hypotheses

Sub-Research Question 1

How can discipline-specific design constraints in both mechanical and electrical engineering domains be classified?

Sub-Hypothesis 1

Design constraints in both domains can be classified through analyzing mechatronic products to identify and classify discipline-specific constraints based on associated function, physical form, and system behavior. Constraints based on functions involve the use of equations, constraints based on form involve observation of the physical appearance, and constraints based on behavior involve analyzing system response. These constraints allows direct mapping of design aspects between mechanical and electrical domains of the mechatronic system.

In Chapter 3 various components commonly seen in mechatronic systems are described, followed by a discussion of different mechanical and electrical constraints. These constraints are then categorized based on functions, form, and behavior. As mentioned in Chapter 1, this research hypothesis aims at establishing fundamentals in search for connection of mechanical constraints to the electrical side, and vice versa, which is a research area that hasn't be well developed.

In this thesis the focus is placed on constraints based on function. In constraint model and implementation of constraint propagation it is shown that defining constraints based on functions allows direct mapping between attributes of mechanical components and electrical components.

Sub-Research Question 2

How can both discipline-specific and cross-disciplinary constraints be modeled?

Sub-Hypothesis 2

Both discipline-specific constraints and cross-disciplinary constraints can be modeled using SysML. SysML supports the specification and analysis aspects, the constraints in the mechatronic system are well represented using SysML's Parametric Diagram.

In Chapter 4 both UML and SysML are shown to have the capability to model the components in mechatronic systems. SysML also has the capability to model parametrics for engineering analysis while UML lacks such capability. Hence SysML is the candidate chosen for modeling constraints in mechatronic system.

It is shown through the various scenario examples in Chapter 5 that the constraint model created based on SysML is successfully implemented to propagate design modifications between a MCAD system and an ECAD system.

Development Question

How can constraint propagation be implemented in SolidWorks and EPLAN?

Development Hypothesis

With the macro capability of both CAD systems, component configuration can be defined with given attribute ranges. A constraint manager can be developed to take the new attribute value resulting from a design modification, calculate the new corresponding attribute value on the other domain based on the predefined constraint relationship in the constraint model, and update the designers.

In Chapter 5 implementation is shown for propagating design modification between SolidWorks and EPLAN. The constraint propagation flow diagram identifies

four case scenarios. The implementation shows successful testing of all four case scenarios. Hence the constraint manager described in the development hypothesis has been successfully developed.

Research Question

How can mechatronic systems be designed in an integrated fashion in order for designers of both electrical and mechanical engineering domains to automatically receive feedback regarding design modifications made on either side throughout the design process?

Hypothesis

Integrated design of mechatronic systems can be realized through the integration of mechanical and electrical CAD systems. One approach to achieve this type of integration is through the propagation of constraints. Cross-disciplinary constraints between mechanical and electrical design domains can be classified, represented, modeled, and bi-directionally propagated in order to provide automatically feedback to designers of both engineering domains.

In the thesis, the primary Research Question and Hypothesis are addressed in the following structure:

- Chapter 3 addresses the classification of constraints.
- Chapter 4 addresses the representation and modeling of constraints.
- Chapter 5 addresses the propagation of constraints through implementation and illustration of propagation scenarios.

The implementation in Chapter 5 successfully runs the test scenarios. The implementation is based on the constraint model developed in Chapter 4. The constraint model uses the constraints identified in Chapter 3. Through three phases the Hypothesis answers the Research Question.

6.2 Research Contributions

- Literature review on state-of-the-art information exchange techniques and identifying research gaps. Major techniques that were evaluated are:
    - Product Data Management (PDM)
    - Formats for data exchange (IGES, STEP, and others)
    - NIST Core Product Model
    - Multi-Representation Architecture
    - Constraint Linking Bridge
    - Active Semantic Networks

    Major research gaps identified are:
    - The number and complexity of existing standards.
    - The need to for developing interacting interfaces for integration of software.
    - The lack of research attention towards communication between mechanical and electrical domains in designing mechatronic systems.
- Classification of mechanical and electrical constraints in mechatronic systems.
    - Constraints based on function.
    - Constraints based on form.
    - Constraints based on behavior.
- Develop an approach for modeling and propagating constraints.
    - A constraint model based on the concepts in SysML.
- Implement a software prototype to validate the approach.
    - Constraint propagation flow diagram.
    - Testing of four case scenarios.

The Value of this Thesis

The dream I envisage is the integrated design of mechatronic systems, in which design modifications are propagated to all domains and relevant product documentations are automatically updated. The value of this thesis is the laying of foundations towards achieving this dream by accomplishing a first step – by achieving the propagation of changes made to CAD models between MCAD and ECAD systems and updating the designs in both mechanical and electrical engineering domains.

Currently there is not much research work that aim at achieving the objective stated in this thesis: propagation of CAD model changes. As discussed in Chapter 2, there are many state-of-the-art technologies developed for information management and systems integration, but they are not candidates for use to propagate CAD model changes. It is assuring to see that organizations such as NIST have the similar vision of this particular research gap and have developed solutions in trying to address this issue, such as developing new parts to STEP to address constraints and parameterization.

As discussed in the Research Vision of Chapter 1, the goal is to achieve fully automated creation of product documentation for mechatronic systems. This thesis is the ground break work that addresses the early stage of achieving such goal: integration between two domains (mechanical and electrical) and automation (partial) on a small clearly defined context.

Future work is expected to be built upon the work done in this thesis as they progress in the direction of achieving the dream in the big picture, that is, to build upon the constraint modeling and propagation approach of this thesis to achieving integration of other domains and to achieving greater scale of automation.

6.3 Limitations

- A constraint relates various parameters of the mechatronic system; however, a parameter may be constrained by multiple constraints. As the number of parameters and constraints increase (i.e., through increasing number components in the mechatronic system), the constraint network may become too complicated and too difficult to actually implement the propagation.

  For example, the motor torque in the robot arm depends on the parameters of weight and moment arm, when any of the parameter is changed, the motor torque is calculated. However, if the motor torque is changed, which of the parameters (weight and moment arm) is affected and which remains unchanged.

  This issue has not been addressed in the thesis. One possible solution to this issue is to develop a more systematic constraint propagation mechanism, and one possible way of doing so is to incorporate existing constraint solving and programming algorithms into the approach discussed in this thesis.

- The list of constraints in Chapter 3 covers fundamental constraints seen in mechatronic systems. The list of categories needs to be expanded to cover a broader variety of cross-disciplinary constraints. In addition, the concept classification based function, form, and behavior needs to be more developed and more examples are needed to support the concept.

  This thesis attempts to address constraints on both conceptual design level and system realization level. One major challenge is that both levels are on different levels of abstraction and hence concepts developed for one level may or may not be applicable for the other level.

- The current implementation only shows communication of software on a single computer. But in reality, the software would be located on multiple computers; hence networking would be a major issue. Expertise in computer science is required to sufficiently address this issue.

  With the current implementation approach of creating the constraint model in an Excel document, a simple solution would be storing that particular Excel document in a network sharing folder so that both mechanical and electrical engineers can access that document as long as their computers are connected to that particular network.

- Information management from database perspective. E.g., user authorization, locking mechanisms for access CAD data to prevent simultaneous and conflicting design modification, release management, connectivity to PDM/PLM systems (integration into overall CAE environment)

6.4 Future Work

- Research on incorporation of principles of constraint programming.

  Constraint programming can be considered as an alternative approach to programming that relies on a combination of techniques that deal with reasoning and computing (Krzysztof 2003). There are numerous constraint programming methods, for instance, constraint logic programming (CLP), is one of them. CLP in essence provide only one programming construct: rules, which allow programmers to define their own constraints in terms of the underlying constraint domain of the CLP language (Marriott et al. 1998). CLP can be used to model and solve various circuit

problems. Hence, it is possible to use CLP in constraint modeling and propagation in mechatronic systems.

- Development of mechanism for data access and control.

  In the current software prototype described in this thesis, there is no mechanism to prevent the designers from making the modification at the same time. The mechanical engineer may be making a change to the MCAD model while the electrical engineer is also accessing the ECAD model to make a change. This can cause major consistency problem. One possible solution is to develop mechanisms for data access and control. Analogous to books in a library, when a book is check out by one reader, all other readers cannot read that particular book. Similar, when a mechanical engineer "checks out" the MCAD model to make a modification, the electrical engineer should not have access to that corresponding ECAD model.

- Expanding the list of constraints and further develop concept of constraint classification.

  Currently the constraints addressed in this thesis are static constraints, so one of the next steps is to address performance related constraints such as stiffness or time lag and how they can be classified and considered in constraint modeling and propagation.

- Address standardization issue: how can this proposed approach be embedded into existing techniques for information exchange in order to make it available to a broader community. For example, can this approach be applied to all commercially available MCAD and ECAD systems? Can techniques described in Chapter 2, such as STEP, Multi-Representation Architecture, be incorporated into this approach discussed in this thesis?

REFERENCES

Ahn, A., Agonafer, D. and Novotny, S., 2004, "Methodology for an integrated (electrical/ mechanical) design of PWBA," *Journal of Electronic Packaging*, **126.**(4.), pp. 524–527.

Appuu Kuttan, K. K., 2007, *Introduction to mechatronics*, Oxford University Press.

Balmelli, L., 2006, "An Overview of the Systems Modeling Language for product and systems development", http://www.ibm.com/developerworks/rational/library/aug06/balmelli/(08/02/08)

Biswas, A., Fenves, S. J., Shapiro, V. and Sriram, R. D., 2008, "Representation of heterogeneous material properties in the core product model," *Engineering with Computers*, **24.**(1.), pp. 43–58.

Bolton, W., 2003, *Mechatronics: electronic control systems in mechanical and electrical engineering, 3rd Edition*, Pearson Education Limited.

Bullinger, H. J., Warschat, J. and Fischer, D., 2000, "Rapid product development – an overview", *Computers in Industry*, **42.**(2.), pp. 99–108.

Centikunt, S., 2007, *Mechatronics*, John Wiley and Sons.

Chen, K. and Schaefer, D., 2007, "MCAD – ECAD Integration: Overview and Future Research Perspectives," *2007 ASME International Mechanical Engineering Congress and Exposition (IMECE07)*, Seattle, Washington, November 10-16, 2007.

Choi, G. H., Mun, D. and Han, S., 2002, "Exchange of CAD Part Models Based on the Macro-Parametric Approach," *International Journal of CAD/CAM*, **2.**(1.) pp. 13–21.

DXF Reference, 2008, *AutoCAD 2009*, v.u.23.1.01

Electronic Industries Alliance, 2005, "About EDIF," *Internet Archive: Wayback Machine*, http://web.archive.org/web/20051218222249/www.edif.org/about.html

Evans, R., 2007, "ECAD-MCAD design integration – smoothing the way," *Altium Limited Newsletters*, http://www.altium.com/files/pdfs/ECAD-MCAD.pdf

Fenves, S. J., Sriram, R. D., Subrahmanian, E. and Rachuri, S., 2005, "Product information exchange: practices and standards," *Journal of Computing and Information Science in Engineering*, **5.**(3.), pp. 238–246.

Fowler, M., 2004, *UML Distilled Third Edition: A Brief Guide to the Standard Object Modeling Language*, Addison-Wesley.

Friedenthal, S., Moore, A. and Steiner, R., 2008, *A Practical Guide to SysML: The Systems Modeling Language*, Elsevier Inc.

Fulton, R.E., Ume, C., Y., F.C., Peak, R.S., Scholand, A.J., S. S., Stiteler, M., Tamburini, D.R., Tsang, F. and Zhou, W., 1994, "Rapid thermomechancial design of electronic products in a flexible integrated enterprise," *Interim Report, Manufacturing Research Centre*, Georgia Institute of Technology, Atlanta.

Hawkinson, C., 2006, "Solving the 5 key challenges of electromechanical product development," http://www.ptcworld.org/Downloads/Asia/EMPD_Hawkinson.pdf

Hoffman, C. M. and Joan-Arinyo, R., 1998, "CAD and the product master model," *Computer-Aided Design*, **30.**(11.), pp. 905–918.

ISO, 2001, ISO/CD 10303-108: "Product data representation and exchange. Integrated application resource: Parameterization and constraints for explicit geometric product models," Committee Draft (CD), ISO TC 184/SC4/WG12/N940, Geneva, Switzerland.

Kim, I. and Lee, S., 2005, "Development of an interoperability model for different construction drawing standards based on ISO 10303 STEP," *Automation in Construction*, **14.**(5.), pp. 633–649.

Klein, R., 1998, "The role of constraints in geometric modeling," In Bruderlin, B. and Roller, D. (Eds), *Geometric Constraint Solving and Applications*, Springer-Verlag, pp. 3–23.

Kleiner, S., Anderl, R and Grab, R., 2003, "A collaborative design system for product data integration," *Journal of Engineering Design*, **14.**(4.), pp. 421–428.

Krzysztof, R. A., 2003, *Principles of Constraint Programming*, Cambridge University Press.

Lee, S. H. and Jeong, Y. S., 2006, "A system integration framework through development of ISO 10303-based product model for steel bridges," *Automation in Construction*, **15.**(2.), pp. 212–228.

Liu, D. T. and Xu, X.W., 2001, "A review of web-based product data management systems," *Computer in Industry*, **44:** pp. 251–262.

Lubell, J., Peak, R., Srinivasan, V. and Waterbury, S., 2004, "STEP, XML, and UML: Complementary Technologies," In: *Proceedings of ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. **4:** pp. 915–923.

Marcheix, D. and Pierra, G., 2002, "A survey of the persistent naming problem," *Proceedings of the Symposium on Solid Modeling and Applications*, pp. 13–22.

Marriot, K. and Stuckey, P. J., 1998, *Programming with Constraints: An Introduction*, The MIT Press.

Mocko, G., Malak, R., Paredis, C. and Peak, R., 2004, "A knowledge repository for behavior models in engineering design," In *Proceedings of ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. **4:** pp. 943–952.

Mun, D., Han, S., Kim, J. and Oh, Y., 2003, "A set of standard modeling commands for the history-based parametric approach," *Computer-Aided Design*, **35.**(13.), pp. 1171–1179.

Nagel, R. L., Vucovich, J. P., Stone, R. B. and McAdams, D. A., 2008, "A signal grammar to guide functional modeling of electromechanical products," *Journal of Mechanical Design*, **130.**(5.), pp. 1–10.

OMG, 2006, "OMG Systems Modeling Language Tutorial"
http://www.omgsysml.org (07/28/08)

OMG, 2007, OMG Systems Modeling Language (OMG SysML)
http://www.omgsysml.org (08/21/08)

Peak, R. S., Fulton, R. E., Nishigaki, I. and Okamoto, N., 1998, "Integrating Engineering Design and Analysis using Multi-Representation Architecture," *Engineering with Computers*, **14.**(2.), pp. 93–114.

Peark, R. S., Burkhart, R. B., Friedenthal, S. A., Wilson, M. W., Bajaj, M. and Kim, I., 2007, "Simulation-Based Design Using SysML, Part 1: A Parametrics Primer," *INCOSE Intl. Symposium*, San Diego.

Pender, T., 2003, *UML Bible*, Wiley Publishing Inc.

Penoyer, J. A., Burnett, G., Fawcett, D.J. and Liou, S. –Y., 2000, "Knowledge based product life cycle systems: principles of integration of KBE and C3P," *Computer-Aided Design*, **32.**(5.-6.), pp. 311–320.

Philpotts, M., 1996, "An introduction to the concepts, benefits and terminology of product data management", *Industrial Management and Data Systems*, **96.**(4.), pp. 11–17.

Pratt, M. J., Anderson, B.D. and Ranger, T., 2005, "Towards the standardized exchange of parameterized featured-based CAD models," *Computer-Aided Design*, **37.**(12.), pp. 1251–1265.

Pratt, M. J., 2005, "ISO 10303, the STEP standard for product data exchange, and its PLM capabilities," *International Journal of Product Lifecycle Management*, **1.**(1.), pp. 86–94.

Pratt, M. J., Kim, J., Iyer, R. G. and Sriram, R. D., 2008, "Standardized data exchange of CAD models with design intent," *Computer-Aided Design*, **40.**(7.), pp. 760–777.

Reeves, B. and Shipman, F., 1992, "Supporting communication between designers with artifacts-centered evolving information spaces," *Computer Supported Cooperative Work: Proceedings of the 1992 ACM Conference on Computer-supported cooperative work*, pp. 394–401.

Rembold, U., Nnaji, B. O. and Storr, A., 1993, *Computer Integrated Manufacturing and Engineering, 3rd Edition*, Addison-Wesley.

Roller, D., Eck, O. and Dalakakis, S., 2002, "Advanced database approach for cooperative product design," *Journal of Engineering Design*, **13.**(1.), pp. 49–61.

Schaefer, D., Eck, O. and Roller, D., 1999, "A shared knowledge base for interdisciplinary parametric product data models in CAD," In *Proceedings of the 12th International Conference on Engineering Design*, pp. 1593–1598.

Schaefer, D., 2006, "A generic approach to variant design in electrical engineering CAD," *ATP International*, (**2.**), pp. 46–50.

Stone, R. and Wood, K., 2000, "Development of a functional basis for design," *Journal of Mechanical Design*, **122.**(4.), pp.359–370.

Sudarsan, R., Fenves, S. J., Sriram, R. D. and Wang, F., 2005, "A product information modeling framework for product lifecycle management," *Computer Aided Design*, **37.**(13.), pp. 1399–1411.

Sung, C. S. and Park, S. J., 2007, "A component-based product data management system," *International Journal of Advanced Manufacturing Technology*, **33:** pp. 614–626.

Ullman, D. G., 2002, *The Mechanical Design Process*, 3rd ed., McGraw-Hill, New York.

Wu, Y., 2003, "Describing integrated power electronics using STEP AP210," MS thesis submitted to Faculty of the Virginia Polytechnic Institute and State University.

Yang, J. and Han, S., 2006, "Repairing CAD model errors based on the design history," *Computer-Aided Design*, **38.**(6.), pp. 627–640.

You, C. F. and Chen, T. P., 2007, "3D part retrieval in product data management system," *Computer-Aided Design and Applications*, **3.**(1-4), pp. 117–125.